63-3-4

[redacted]

Ford Motor Company
AERONUTRONIC DIVISION

Ford Motor Company
AERONUTRONIC DIVISION

Publication No. U-2111

RESEARCH LABORATORY

ANNUAL SUMMARY REPORT

RESEARCH ON BIAX TYPE ELEMENTS AND ASSOCIATED CIRCUITS (BIAX PERCEPTRON)

Reporting Period: 1 February 1962 to 31 January 1963

Prepared for: Office of Naval Research
Department of the Navy
Washington 25, D. C.

Under Contract: NOnr-2913(00)
Amendment No. 4

Prepared by: J. K. Hawkins
C. J. Munsey
R. A. Stafford

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION 1

PROGRAM SUMMARY

This report describes research work whose goal is to develop useful, efficient and economical learning machines. A learning machine may be broadly defined as a mechanical system that behaves in a manner similar to biological systems under conditions in which the latter are described as "learning."

Within this general framework, the problem is to devise networks and corresponding rules of behavior for the network elements which will yield the desired overall behavior. A great variety of network and element types are possible. The work reported here is restricted to networks consisting of linear-input logic elements. The reasons for this selection are discussed in the body of the report.

The previous state of understanding of learning networks consisting of linear-input logic elements may be described as follows. The behavior of single element (one-stage) networks was reasonably well understood. Proofs of convergence to a "learned" condition (absence of errors) for the network existed, and bounds on the number of errors before convergence had been derived.* For multi-element networks, however, only fragmentary evidence of learning behavior existed, and convergence had only been proven in special and rather inefficient cases. In addition, no general guiding principles had emerged.

---

* Appendix A, "A Magnetic Integrator for the Perceptron Program", Annual Summary Report, Aeronutronic Div., Ford Motor Co., Aug. 1961, ASTIA #AD 264227.

The work reported here was confined primarily to the multi-element learning network problem. The major results may be summarized as follows:

(1) Based upon the experimental work reported, three principles which appear to be fundamental to successful multi-element network learning have been formulated.

(2) A consistent body of experimental evidence supporting these principles has been developed.

(3) A parallel logic technique has been developed which makes possible the economical mechanization of large learning networks.

In support of these results, the research work which was performed can be divided into four phases. They are as follows.

In order to provide a standard of comparison for multi-element learning networks, a listing of all possible linear-input logic mechanizations of functions through four input variables was prepared. Based upon this list, functions can be arranged in some order of relative learning difficulty. The function difficulty, and possible forms of mechanization, therefore represents a standard against which experimental results on specific functions and learning methods may be compared. The list of four-input functions is presented in the Appendix.

Experiments on learning in multi-element networks were conducted on the learning machine previously constructed for the Office of Naval Research and Rome Air Development Center.* The machine and experiments conducted on it are described in Section 2. Based upon these experiments, and other simulations, new learning network principles were formulated.

The new principles of network learning were reduced to specific form and hand simulated. The principles may be briefly described as: 1) excess network capacity, 2) network self-evaluation, and 3) least-effort adaptation. Striking success with early hand simulations led to the writing of a computer program for conducting more extensive exploration. The program was completed and run on a variety of multi-element functions,

---

* "A Magnetic Integrator for the Perceptron Program", Annual Summary Report, Aeronutronic Div., Ford Motor Co., Aug. 1961, ASTIA #AD 264227.

including those from the four-input list described above. The results were notably successful. This approach to multi-element network learning, a summary of the computer program, and its results are presented in Section 3.

In view of the practical need for large networks of learning elements, efforts were made to convert a parallel logic technique to this type of operation. A mechanization for large parallel arrays of learning elements was therefore developed. Single-layer convergence of such arrays was proven, and a simple example performed. These results were reported,* and are recounted briefly in Section 4.

* J. K. Hawkins, C. J. Munsey, "A Natural Image Computer", Optical Processing of Information, Spartan Books. (to be published)

SECTION 2

LEARNING MACHINE EXPERIMENTS

Experiments of learning processes on multi-layer networks of adaptive elements were performed upon the machine constructed for the Office of Naval Research and Rome Air Development Center, previously reported.* These experiments provided valuable early insight into the nature of multi-layer network behavior. They also suggested a basis for the improvements in network learning principles which are formulated in Section 3.

In order to provide a self-contained report, this section recapitulates briefly the principles of individual learning elements, and describes the structure and operation of the experimental machine. It concludes with a brief report on the experiments conducted on the machine, which in turn led to the revisions in approach discussed in Section 3.

2.1 MACHINE DESCRIPTION

The machine to be described is a special purpose computer. It was conceived and designed to explore experimentally self-organizing logic, and its realization in working hardware. This machine is unique in that the basic logic modules may be connected in cascade without limit. This freedom permits experimental realization of the important multi-layer logic functions. Another unique feature is the magnetic integrator employed as an adjustable weighting and storage element in the logic module.

---

*"A Magnetic Integrator for the Perceptron Program", Annual Summary Report, Aeronutronic Div., Ford Motor Co., Aug. 1961, ASTIA #AD 264227.

The basic components of the machine are linear threshold logic elements. In this type of logic element, binary inputs are linearly weighted and summed. The binary output (proposition) is the result of a threshold decision on this quantized set.

Given a network of linear-threshold logic devices, it is possible to alter the binary transmission of the network by altering the weight values. This adjustability of the network permits its transmission to be brought into congruence with a desired standard. It has been demonstrated, at least for a single stage, that weight change strategies exist which insure convergence to any realizable transmission function. The machine to be described employs a successful single stage strategy, with the exception that extra controls are added to permit multi-stage experiments.

a. Control of Stored Value Change

The value of magnetic flux stored in the integrator, called the W-unit, represents the present state of the machine and must be changed to obtain new states. This change is brought about by built-in logical rules. The W-units also provide interconnections between the amplifiers (A-units) which perform the threshold decision upon the input sum. The machine, as constructed, has a built-in logic for value growth which may be stated by three rules:

(1) Increment only if "control" is ON.

(2) Increment only if the actual output of the associated A-unit does not match the selected desired response.

(3) Make the sign of the increment equal to the sign of the algebraic product of the input to the W-unit and the output desired from the A-unit.

Rules one and two are implemented in the A-unit, but rule three depends on both the A and W units.

b. Machine Structure

The computer contains 32 linear-threshold logic units and the necessary supporting apparatus. This includes input, programming, control and power supply sections. All equipment is contained in a pair of six foot racks as illustrated in Figure 2.1 and 2.2. At the upper left are

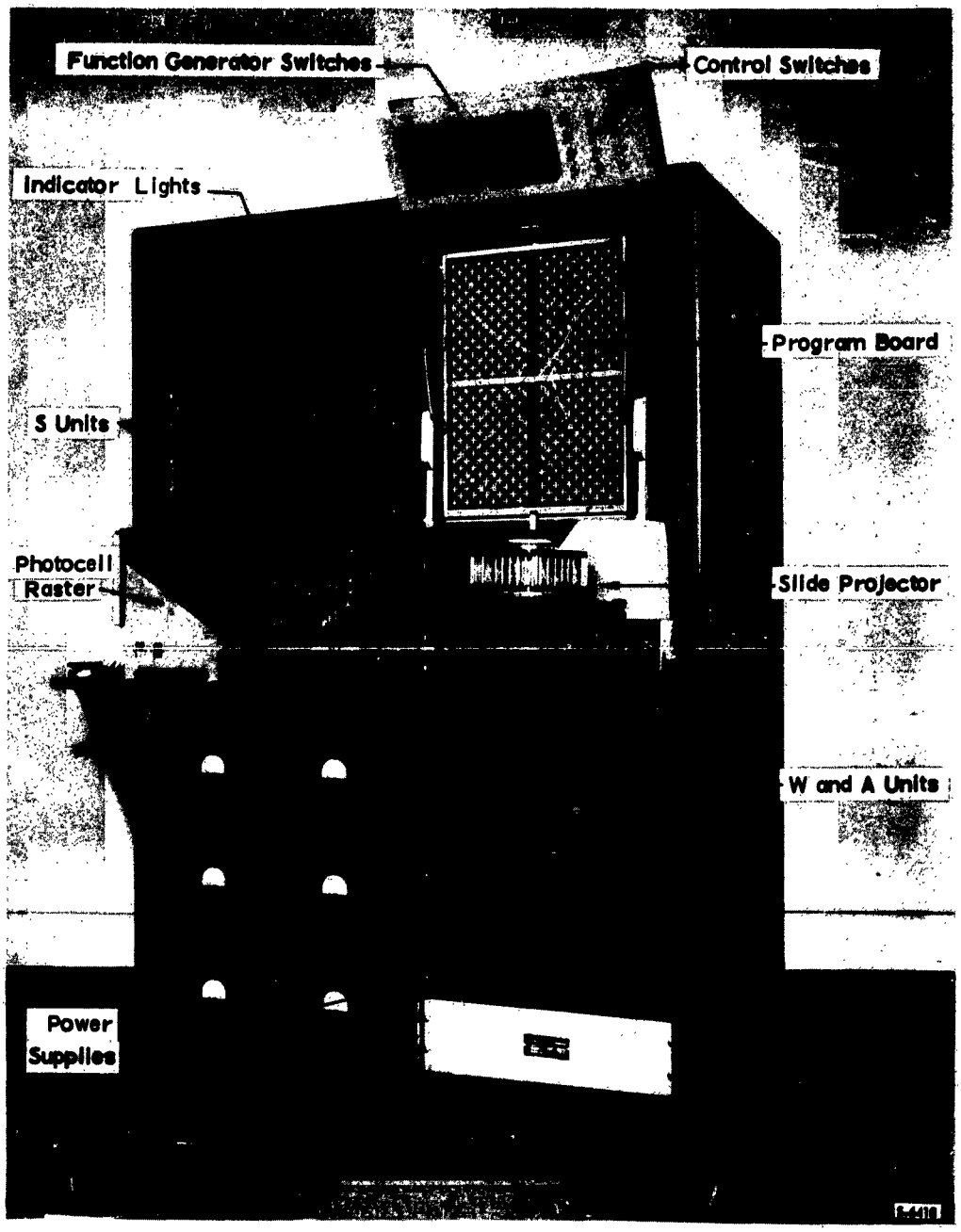


FIGURE 2.1. COMPUTER - FRONT VIEW

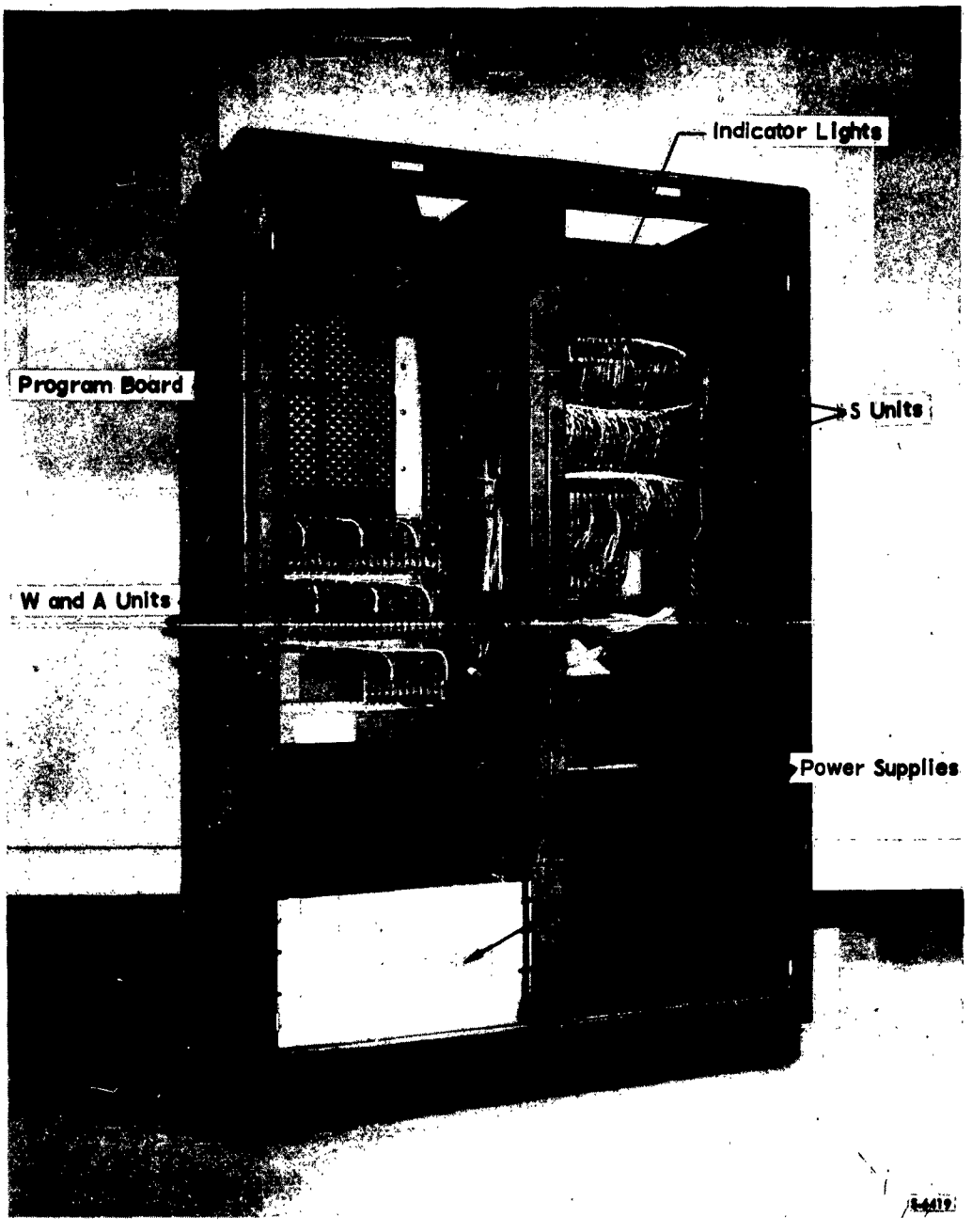


FIGURE 2.2. COMPUTER - REAR VIEW

located several square arrays of indicator lamps. These are used to display the state of the input (when it is an image) and, through the Program Board, selected internal states.

An image input is provided. This takes the form of a 16 x 16 bit raster of photoconductors, each connects to an individual flip-flop (S-unit). In effect, the S-unit converts a change at the photoconductor into a binary signal. Light images are projected onto the raster with an automatic 35 mm slide projector.

An alternate input apparatus consists of a five stage binary counter to provide, in rapid succession, all combinations of up to five binary variables. In conjunction with this counter is a conventional gate and switch arrangement which permits any of the functions of five variables to be formed.

The Program Board, a removable patch board, contains the leads necessary to connect the linear-logic units (W-A units) into a desired network configuration.

Each threshold-logic unit consists of two circuit boards. One board contains up to eight magnetic integrator weighting devices and associated transistors. The A-unit contains the remainder of the threshold logic and self-organizing functions.

c. Operation

To operate the machine the experimenter makes a set of connections on the Program Board to set up the desired network. In addition, he chooses several other variables concerned with the succession of events which control the weight changing (self-organizing) behavior.

Each network element requires connections, in addition to input and output, for increment control, and to signal when a desired response has occurred. A variety of possible sources or destinations for these signals are presented in Table I.

A given network is tested by exposing it to a sequence of different inputs; an input being a binary word of arbitrary length. Each input is accompanied by a signal which indicates the network response desired. When the response is incorrect, weight values are altered (incremented) according to the given rules.

The experimenter has some weight changing options which he must exercise. These are:

TABLE I

SIGNAL	POSSIBLE SOURCE OR TERMINATION
Input	1. Internal pattern counter. 2. Photocell raster (S-units). 3. Output of other A-units.
Output	1. External indicator (Oscilloscope) 2. Input of other W-units. 3. Desired response or increment control of other A-units.
Desired Response	1. Desired output flip-flop, associated with the internal pattern counter. 2. Certain photocells in the input raster which transduce a code word placed on the input slide. 3. The output of other A-units.
Increment Control	1. Bank counter. 2. The output of other A-units.

(1) From one to sixteen increment actions may be taken each time an input set is presented. This choice is made by appropriate settings of some of the "Control Switches" shown in Figure 2.2.

(2) The network may be divided into sections for increment action. Thus, incrementing proceeds normally in a chosen section while the weights on all other sections remain fixed. This option is controlled by the choice of signals for the logic units "increment control" input.

In conjunction with item two, a special source of increment control signals is called the Bank Counter. It provides from one to four time periods for dividing the network into as many as four separate increment sections.

2.2 EXPERIMENTS

Two main types of multi-element learning experiments of interest were conducted in a variety of configurations during the reporting period. Both groups, each corresponding to a specific network structure and learning rule, failed to yield satisfactory learning behavior. However, they provided useful insight into the requirements for learning rules in such networks.

The first group of experiments involved a simple cascade network of the type illustrated in Figure 2.3. The element labelled Y was given the usual error-correcting learning rule. The element, X, however, was given the error correcting rule only when Y was not correct. Otherwise, it was not changed. If Y ever becomes correct for all input configurations, the network is stable. That is, if Y is correct, neither the weights into X nor the weights into Y are allowed to change. If Y is correct over all entries in the truth table, no further changes occur.

If Y is incorrect for any entry in the truth table, both X and Y input weights are allowed to change. In all cases, Y was corrected according to the desired output. Several versions of change rules for X were tried. One was simply to let X be corrected according to the desired output. Another was to let it be corrected according to the complement of its (X's) present output. Still another was to alternate corrections, first letting only X inputs change, then Y inputs, then X inputs, etc.

This circuit arrangement and the above rules were tried on a number of examples of four-input functions known to be realizable a) by a

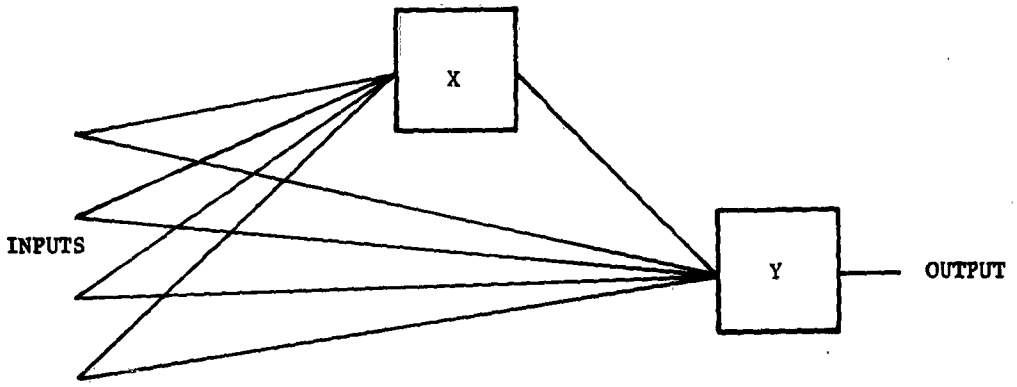


FIGURE 2.3. TWO ELEMENT CASCADE

single threshold element (such as Y alone), or b) by two threshold elements arranged as in Figure 2.4. The particular functions tried were selected from the list of classes of four-input functions in the Appendix. The first seven classes are realizable by single elements. Of those requiring at least two elements, representatives from classes 8, 9 and 10 were selected. These may be regarded as the "easiest" two-element classes to realize, in the sense that greater variety is possible in the function realized by the first element in the cascade, while still realizing the overall function.

As might be expected, the network converged to correct behavior under condition (a). It is interesting to note that element X, as might be expected, frequently took advantage of the limitations on its change rule to realize a function which was only a poor approximation to the desired. In many cases, in fact, it produced an ambiguous output (sometimes 1, sometimes 0) for some input configurations. These same input configurations therefore represented the determining factor in the Y element output, the X-to-Y input corresponding to a "don't care" condition.

When tried under condition (b), however, the network consistently failed to converge for even the simplest two-element functions. This is rather surprising, since it can be shown that for the very simplest two-element function (class #8), a random selection of weights on element X yields approximately a 50-50 chance of picking a function which permits Y to realize the desired function.

The second group of experiments employed the structure of Figure 2.4, in which element Y was preset to be a fixed (nonlearning) 2-out-of-3 majority function. Elements X were then treated in the same manner as their counterparts in the first group of experiments. Again, both (a) single element and (b) two-element functions were tried. In case (a) the structure always converged. In case (b), only some representatives of the three simplest types of cascade functions would converge, and then only with great apparent difficulty. The difficulty was indicated by the long time required for convergence (several seconds), and by the ease with which the function could be "unlearned" due to the presence of minor noise signals. Even after a function is learned and the network behaves correctly for some period of time, a change in signal level can create a mistake. If this occurs, the system is automatically incremented, thus disturbing an obviously delicate set of weight values. A long and difficult repeat learning sequence then ensues.

This network behavior pointed up the need for a mechanism in the learning rule which would cause first-layer elements to diverge in

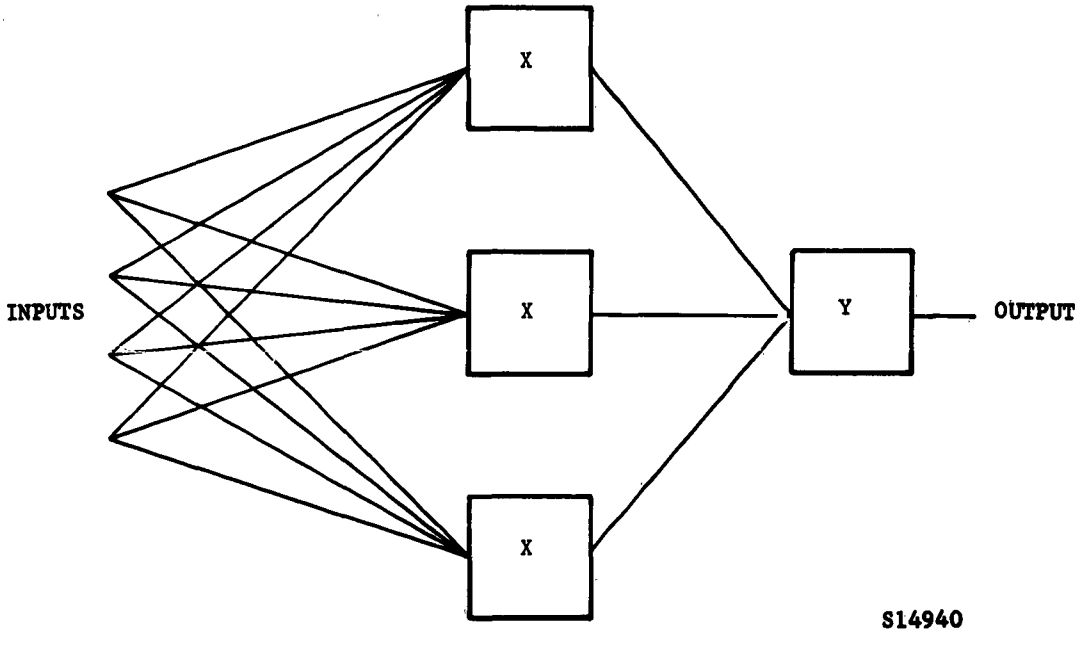


FIGURE 2.4. MULTI-ELEMENT CASCADE

terms of the functions they are realizing. The effect of both of the above groups of experiments is clearly the opposite, since in all cases, first-layer elements can only tend toward one another.

SECTION 3

SIMULATIONS OF LEARNING NETWORKS

The developments reported in this section depart in a number of ways from accepted concepts in the field of learning machines. These departures have been brought about by analyses of single threshold element learning behavior, by experiments performed on the magnetic integrator learning machine, by numerous hand simulations, and by results from a computer simulation program for an IBM 7090 computer.

Based upon these research efforts, a number of basic learning network principles have been formulated. All indications point to the fact that any learning network must conform in some manner to these principles if the network is to fulfill the requirements placed upon it. The remarkable successes of a variety of learning network simulations which make use of these principles constitute powerful evidence of their general validity, and warrant further investigation.

It should be noted that the principles described herein were strongly influenced by the practical problem of avoiding overly complicated rules for network weight changes. This has led to the important concept of using the computational capacity inherent in a network (i.e., its ability to determine an output when given an input) for greatly assisting in determining how its weight changes may best be made during the learning process.

This section is organized as follows. Section 3.1 describes and defines the properties required of a learning network. Properties of single learning elements are briefly discussed in 3.2, and some of the early attempts and difficulties with multi-element networks are recounted in 3.3. Section 3.4 presents general principles which have come out of these earlier attempts and which represent the most recent advances.

Finally, Sections 3.5 and 3.6 describe a computer simulation program using these principles and the encouraging results obtained from it so far.

3.1 PROPERTIES REQUIRED OF A LEARNING NETWORK

The term "learning network" is applied to a system possessing the following general properties. First, there is some form of input consisting of a number of variables, $x_1, x_2, \ldots, x_n$, each of which may be a discrete valued or continuous valued numerical quantity. In the extreme case they may be binary valued in which case the system can be described as a logic system. Each combination of input variables is, of course, regarded as different from another if any of its variables differ from the corresponding variable of the other. Thus, for n binary valued variables there are 2^n distinct combinations. The term "input" will hereafter be understood to refer to a combination of input variables.

Second, there is to be an output consisting of one or more variables, $y_1, y_2, \ldots, y_m$. Remarks made above for input variables apply also to outputs. The outputs are to be produced by the network in response to given inputs according to some rule which is determined by the network structure and its internal parameters. Much of the discussion below deals with an output consisting of a single binary valued variable. This is both because of the greater simplicity of such a network, and because placing m such systems together with the same set of inputs could yield the more general m variable output system.

Third, the network is to be capable of responding to a second kind of input which can be called an error signal. This response is to take the form of altering certain of its internal parameters in such way that the rules determining output as a function of input are altered. The change that is made to the parameters of the network must, of course, be aimed at achieving the desired output. After being presented with the various possible inputs and required error signals with a sufficient number of repetitions, a successful network will have "learned" to respond to each input with the desired output.

Fourth and last, the system should be composed of a number of elements of simple and more or less uniform properties. Each of these elements should be able to accept a certain number of input variables and to produce an output variable in accordance with some changeable internal parameters in the element. They should be interconnected by allowing the outputs of some to be used as the inputs of others (in addition to the basic input variables, $x_1, x_2, \ldots, x_n$), the outputs of these in their

turn being inputs to still other elements, etc. Moreover, the specification for this scheme of interconnection should be a relatively simple one. That is, the network, though possibly containing a very large number of elements and interconnections, ought to be relatively simple to characterize or specify. This allows the design of such a network to be fairly uncomplicated, merely involving a large number of repetitions of a basic scheme.

3.2 SINGLE LINEAR THRESHOLD ELEMENTS

The inspiration for the concept of a learning network owes much to the known or postulated properties of the human central nervous system. The latter constitutes the best example of such a network. The neuron with its numerous inputs from other neurons or from sensory inputs, and its single output constitutes a fine example of the postulated element. At an early stage many workers in this field arrived at the concept of a linear threshold element by abstracting certain logical properties of neurons. Though it is in no sense a copy of a neuron, it has shown itself to be a good candidate for the postulated element of a learning network.

Let $x_1, x_2, \ldots, x_n$ be two-valued variables whose values we assume for convenience are +1 and -1. Using these as inputs to a linear threshold element, it is to possess a set of coefficients or weights, $c_0, c_1, \ldots, c_n$ and the means to form the linear combination, $s = c_0 + c_1x_1 + \ldots + c_nx_n$. If $s \geq 0$ its output is to be $y = +1$, and otherwise $y = -1$.

There are a number of different schemes for altering the weight values when it is desired that the output change. The one used here is that, for inputs $x_1, \ldots, x_n$, if the output, y, is not equal to y*, the desired output, then the weights should be changed according to the equations, $\Delta c_0 = y^*\delta$, $\Delta c_i = y^*x_i\delta$, $i = 1, 2, \ldots, n$, where δ is some positive value. If on the other hand $y = y^*$, no change should be made.

It has been realized by several groups independently that such an element taken by itself has many of the required properties for learning networks. It can be proven that if a logical function can be realized in a single such element with some set of weights, then the above weight change procedure is guaranteed to make no more than some finite number of errors before the function is learned completely. A proof of this convergence is contained in the previous Annual Summary report prepared under this contract, "A Magnetic Integrator for the Perceptron Program." An element of this type also exhibits a pronounced tendency to produce like outputs for similar input combinations. Finally the sum, s, gives a

rough measure of how large a change of weights need be to correct a given output (a fact of great importance as will be explained later.)

However a single threshold element has a fatal shortcoming when used as the entire learning device. It does not have sufficient flexibility to assume by itself even a reasonable proportion of all possible logical functions. For example, less than one in each trillion logical (Boolean) functions of six inputs is one which can be obtained by some set of weights in a linear threshold element.* This ratio gets worse as the number of input variables increases. Moreover, on many of those functions which are theoretically capable of being learned by a single linear threshold element, the number of repetitions of errors to be expected in the learning process is absurdly large. For example, well over half the linear threshold functions of six inputs can be expected to produce of the order of 500 to 1,000 errors before the function can be learned, despite the fact that only 64 input combinations are involved. (In fact, most of these errors will involve only about a dozen of these inputs, as a rule.) An upper bound on the number of corrections required starting with zero weights for convergence of linear threshold elements through six inputs is listed in Table II.

However, it can be easily demonstrated that any logical function can be produced from a set of inputs if a sufficiently large network of these linear threshold elements is used. Because of the promising properties of such networks, nearly all work under this contract has presupposed a network of linear threshold elements, and has endeavored to find a suitable rule for making weight changes within the network. The next subsection describes this endeavor.

3.3 EARLY INVESTIGATIONS OF WEIGHT CHANGE RULES

For any single linear threshold element, if it is known what output is desired for it, there is no difficulty in so altering its weights that the proper output is achieved. However, in a network with many such elements, the error signal tells only that the output of the final element is incorrect (in the one-output-variable type of network.) There is no direct way of telling from this fact what the appropriate weight changes are for each of the elements, because except for the final element it isn't clear what their outputs ought to be.

---

* There are 15,028,134 functions of six inputs which can be generated by such an element, against 2^{64} altogether. Table I is a list of the relative number of linear threshold functions for up to six inputs.

TABLE I

RELATIVE NUMBER OF LINEAR THRESHOLD FUNCTIONS

n	Number of Linear Threshold Functions (N_T)	$N_T/2^{2^n}$
1	4	1.0
2	14	.88
3	104	.41
4	1,882	$.29 \times 10^{-1}$
5	94,572	$.22 \times 10^{-4}$
6	15,028,134	$.31 \times 10^{-12}$

TABLE II

MAXIMUM NUMBER OF CORRECTIONS FOR CONVERGENCE

n	Corrections
2	6
3	28
4	95
5	354
6	1421

Moreover, the rule of decision for each individual element as to whether and in which direction its weights should change in case the final element is in error should be a relatively simple one not involving the action of other elements in the network except those which are its inputs. This is necessary in keeping to the concept of simple network elements. It would be absurd to construct elements with limited logical capabilities but nevertheless requiring very complex decisions as to weight changes.

A scheme which was tried early and which satisfies this simplicity criterion is the following. If the final element is in error for a given input, commence to gradually change the weights of each element (including the final one) whose output is not equal to the desired output for the final element in accordance with the rule given in Section 3.2 for single elements, considering the desired output of the final element as also that desired for the element being altered. Let this continue until the final element gives the correct output. Note that some of the elements whose weights are being changed may not have reversed their outputs by the time this occurs if their sums, s, were originally large. In any case when the final element is not in error all weight changes stop. Under this rule, each element needs to possess only an indication of the final output value desired, the fact that it is or isn't being realized, the states of inputs into the element, and its own output. The hope in this scheme was that by tending to force the outputs of the elements towards that desired by the final element whenever the latter is in error, each element would assume some responsibility for a portion of the truth table of the desired logical function.

However, this method was found to be a total failure in hand simulations, failing even for some logical functions of only three input variables. In experiments described in Section 2.2 on the magnetic integrator machine on non linear threshold logical functions, the results were apparently worse than if the weights had been altered in a completely random way each time an error occurred! This perverse behavior was apparently due to a pronounced tendency in this method for all elements to act more or less alike rather than to diversify their action as was hoped. The result was that such a network would learn linear threshold functions excellently and very little else. The hopelessness of this scheme was clearly demonstrated when by chance the desired function would be nearly learned with only a single error in its truth table remaining. Upon presentation of the input combination which would produce this error, the usual result was that the weight changes made to correct it would wreak great havoc to the remainder of the truth table, and one would be as far away from a solution as ever.

In an attempt to remedy this latter difficulty several variations of the scheme were attempted. One was to follow the above method except that no change was to occur for an element whose sum exceeded a certain bound. This was an attempt to restrict weight changes to a smaller number of elements and in particular to those requiring only small weight changes to reverse their outputs. This attempt also failed. If the bound was made high the network behaved as before. If the bound was low many errors occurred which went uncorrected by virtue of the fact that all sums in these cases were in excess of the bound.

Another idea was to proceed as in the first method except that in case of error a single element would be selected at random for weight change from among those whose output disagreed with the desired final output, rather than changing all such elements. Hand simulations showed that this method too, was far from being successful. However, it is significant that a great improvement did occur with this method. For example, when only a few errors remained for a function there was a much smaller chance of the above-mentioned return to chaos upon occurrence of these errors. It was the discovery of this improvement which was instrumental in leading to one of the principles to be described later. Aside from the fact of not being successful, this method also suffers from the requirement for a random action, something which can prove difficult to mechanize in the proper way.

Another attempt was made which abandoned the notion that each element should tend toward the desired final output in case of error. Instead, whenever an error occurred an element was selected at random to have its weights changed so as to reverse its output. This would continue until the final element reversed its output, either because it was eventually so selected for weight change or because its inputs had changed suitably. Again hand simulations showed that this method was not satisfactory.

All of these methods had one common failing. Far too many weight changes occurred which, instead of helping to correct the final element's output, actually worked against it. The situation was analogous in a way to a servomechanism whose error feedback does not indicate the direction of the error, but only its presence. This viewpoint of these failures led to another of the principles to be expounded in the next section.

3.4 PRINCIPLES OF WEIGHT CHANGE

Out of the failures described in Section 3.3 have come weight change rules which show in computer simulations every indication of leading

to a solution to the problem for a one-output learning network. Section 3.6 gives the results of these computer simulations.

As discussed previously, the methods which were tried earlier suffered in that it was not known whether a given change of output for an element in the network would help, hinder, or do nothing at all for the final element. This is because the effect of an element on the final element depends on a chain or possibly many chains of weights encountered as one traces the interconnections from the given element leading to the final element. That is, if element 1 is an input to element 2 with weight c_{12}, 2 an input to 3 with weight c_{23}, etc., and finally n-1 an input to n with weight $c_{n-1,n}$, then the sign of the product $c_{12}c_{23}\ldots c_{n-1,n}$ gives the sense of the effect of a change in element 1 on element n (assuming this is the only such chain from 1 to n.)

If these interconnecting weights (weights on the output of one element used as input to another) are allowed to assume either sign as was previously assumed, it would clearly be impractical for an impending weight change in an element to depend on the signs of a chain, or worse many chains of such interconnecting weights. To do so would be to assume the existence of a network of interconnections among the elements for the purpose of deciding on weight changes far more complicated than the network of input-output interconnections.

There appear to be just two ways out of this difficulty. One is to put a restriction as to sign on all interconnecting weights, and the other is to discover the effect on the final element by some process which experimentally alters the output of the given element. The latter possibility will be discussed later, but for now the former restriction is adopted. It is assumed that a one-variable-output network will admit only of positive interconnecting weights, though weights on the basic inputs, $x_1,\ldots,x_n$ may be of either sign. Such a restriction makes sense out of the previous criterion that changes made to the outputs of elements were to be in the direction of that output currently desired by the final element. Such a change in an element would always tend to change the final element in the proper direction.

It should be noted that this restriction to positive interconnecting weights is not a serious limitation on the logical capabilities of a network. In fact, if a certain logical function can be achieved in a given network which allows interconnecting weights of either sign, then the same function can be obtained in another network with only positive interconnecting weights and, at the very worst, twice the number of elements. This is done in the worst case by generating for each element of the

original network, it and its complement in the new network. Wherever a negative weight occurred on this element as an input to another, the complement can be used with positive weight instead. (It is assumed in this reasoning that there are no "loops" in the network, such as for example, the output of an element being its own input.)

A second difficulty with previous methods involved the fact that correcting the output for a given input resulted in too many weight changes throughout the network. This results in undoing previously learned outputs to an excessive degree. What is desired is some form of weight change which will correct the present output and yet tend to make the least possible change on outputs for other inputs.

It will be noted that the last two methods outlined in Section 3.3 achieved a very rough way of minimizing the weight changes by changing only one element at a time, that one being selected at random, and this process continuing until the final output becomes altered. Even with the above change of allowing only positive interconnecting weights random selections of this kind have continued to produce unsatisfactory though still further improved results. This is apparently due to the fact that in the random selection no account is taken of the magnitude of the sum occurring in the element to be changed. Consequently elements requiring a very large weight change can as well be selected as those requiring a small change. Yet the required weight change tends to be a good indicator of the amount of ensuing change in logical behavior of the element and therefore of the logical properties of the network.

Hence the second principle of weight change rule to be adopted in this section calls for the selection of elements whose sums are close to zero over those with large sums and the minimum number of these necessary to achieve a correction on the final output. Presumably in a large network the total number of elements thus affected would be only a small fraction of those present. In other words even in the presence of errors a change of weight ought to be a relatively rare event for any particular element.*

Two ways of proceeding to this goal will be described here. The first is as follows. In case of error in the final element, among all elements whose output is not equal to that desired for the final element, choose that one whose sum is closest to zero for weight change.

---

* One is tempted to speculate here that neurophysiologists have not been able to detect long-lasting kinds of changes in the logical action of neurons in nervous tissue simply because of the possible scarcity of such changes compared to the number of neurons present.

Change this element's weights as described in Section 3.2 by a magnitude sufficient to reverse its output. If there is still an error proceed again in the same way to choose the next such element whose sum is closest to zero for change. Finally, when enough elements have been changed the final element will have been corrected. It itself may or may not have had a weight change, depending on the size of its sum. Notice that when the output of an element is altered this may change other sums or even other outputs, and therefore this process is not necessarily equivalent to choosing in one step the k least sums where k is the minimum number necessary for correction of the final output.

It will be observed that the criterion described here would appear to violate the rule of simplicity described in the second paragraph of Section 3.3 which requires that the weight change decision for each element should be a relatively simple one not depending explicitly on the state, and in particular on the sums, of other elements save possibly those which are its inputs. Yet if the decision is to change that element whose sum is closest to zero, the decision for any particular element would necessarily depend on the sums of other elements.

However, if a sequential type operation is allowed (and it already is since the elements are being changed one at a time, sequentially), this difficulty can be avoided. In addition to the usual threshold constant, c_o, let each element have a variable input called the bias, b, which however at any instant is the same for all elements. Thus the output of an element would be $y = \text{sign}\,(c_0 + c_1x_1 + \ldots + c_nx_n + b)$. At the initial presentation of an input the bias commences at zero. But in case of error the bias begins to change in the direction which will tend to produce the correct answer (i.e., plus if +1 is desired, and minus if -1 is desired). The first element whose output reverses is automatically that element with wrong output whose sum is closest to zero. If each element has the property that its weights are changed only when an error signal is given and when its sum is in the immediate neighborhood of zero, the element just described will commence to undergo weight change. If the bias b is then gradually returned to zero it will keep this weight change proceeding until the element's output remains reversed even with zero bias. This accomplishes precisely the first of the steps described above and yet no element requires a direct knowledge of other sums for its weight change rule. The decision depends on other sums only indirectly through the actions of the shifting bias level.

A schematic showing the first step in the above process is presented in Figure 3.1. In (a), elements are shown in their original states, being 1 for positive input sums (●) and -1 for negative (0). As the bias is raised, the effect is to lower the sum level at which the -1 to +1

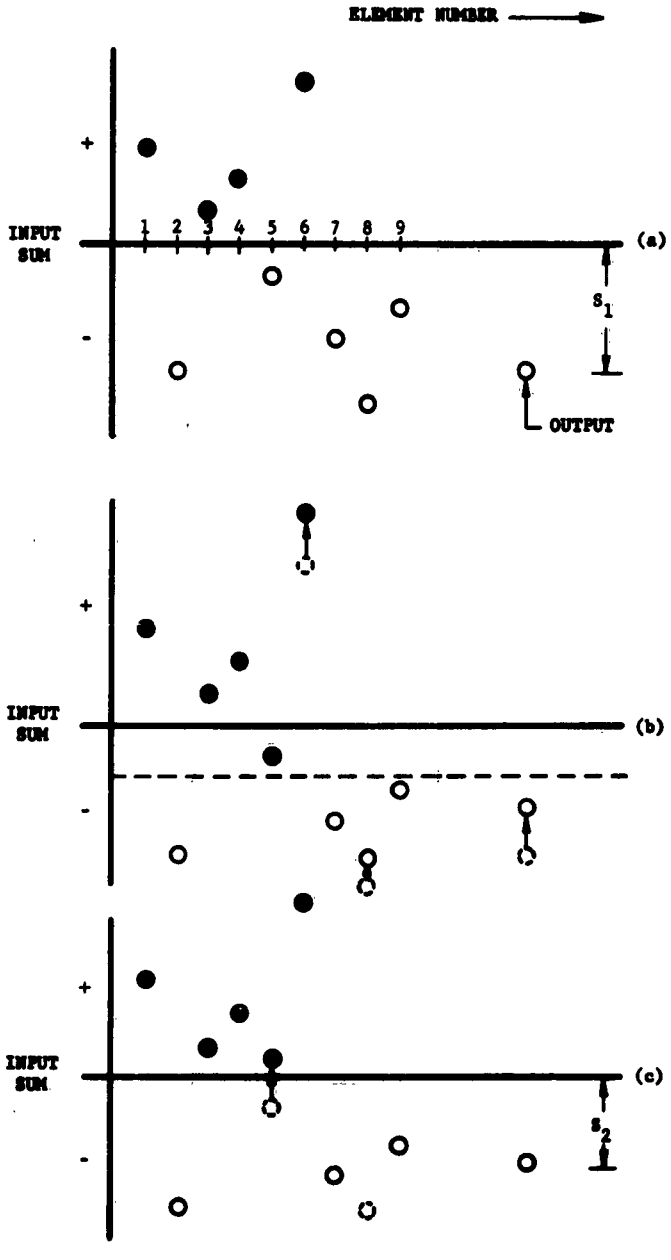


FIGURE 3.1. RISING BIAS SELF EVALUATION

discrimination is made. When the bias crosses element #5, Figure 3.1(b), it becomes a +1. If it is directly or indirectly connected to the output or to other elements, it can only have the effect of driving their sums positively, as indicated. Following the return of the bias to zero, element #5 is permanently (for this input configuration) returned to +1. In particular the absolute value of the output element's sum is guaranteed to be no farther from zero than it was previously. That is, $s_2 \leq s_1$. Clearly, the next element to be changed in this process would be #9, and so on.

The method just described assumes the presence of an agency or unit which generates a common bias level for all elements and which fluctuates it in accordance with the above description. This method has the disadvantage that once an element has reversed, the bias must commence back toward zero. This presupposes that the bias-producing unit has inputs from each of the network's elements if it is to successfully make such a reversal. This again adds an undesirable element of complexity to the network. In order to obviate it a second method is given which produces a similar result without this trouble.

Let the same common bias be provided. Assume that the bias commences with a sufficiently large value of the proper sense so that the final output is initially correct. Then allow this bias to move toward zero. When and if the final output reverses let the error signal then be immediately turned on. Among the elements there must at that time be at least one whose sum lies close to zero and whose consequent reversal caused the reversal of the final output (it could be the final element itself.) If each element has the property that it increments its weights in the presence of an error signal and a sum near zero, the element in question will then undergo weight change. If it is assumed that this increment is a fairly large one rather than the continuous sort envisioned in the previous method, this element's sum will be displaced a considerable distance from zero and its output changed back again. (It is possible that this element will eventually undergo further increments as the bias moves toward zero.) Whenever such an element has been incremented this should immediately correct the final output and thereby turn off the error signal.

As the bias proceeds toward zero many elements may have their sums pass through zero, reversing their and other outputs. As long as this doesn't reverse the final output these elements will avoid being incremented. However, when the final output again reverses the process repeats itself with the output being immediately corrected as some element is incremented. When the bias reaches zero a correct final output will have been obtained.

A schematic outline of one step in this process is presented in Figure 3.2. Initial states are as in Figure 3.2(a). Consider the bias raised to a high value, and now coming slowly down, as in Figure 3.2(b). The output may now be correct, as shown, either due to the bias itself or to the effects of other elements. In the case illustrated, elements 2, 7 and 8 must have caused the indicated shift in the output element's sum. When the bias crosses element #8, Figure 3.2(c), another shift may occur. If insufficient to change the output, however, no incrementing occurs. When the bias crosses element #2, Figure 3.2(d), another shift may occur. In this case the output changes state, and #2 is given an increment. The state of the network immediately after the increment is illustrated in Figure 3.2(e), with the output again correct (temporarily). As the bias falls further, either element #7 or the output itself or both may receive increments to cause the output to remain in the proper state when the bias finally reaches zero.

This then outlines a second method of obtaining a kind of minimal network change to correct an erroneous output. The bias producing agency need not depend on anything other than the desired final output. The error signal depends only on the actual and the desired final outputs. The decision to change weights for each element depends only on the error signal, the bias, and the element's own sum value. This presumably satisfies the criterion stated above for the simplicity of each element's decision rule. Moreover it corrects the output with a single sweep of the bias value whereas the first method requires a sweep for each incremented element.

It should be noted that this second method may have the very important advantage over the first that it selects only those elements to be incremented which have a demonstrable effect on the output. The use of only positive weights prevents element changes which tend to harm the output, but the first method of bias change may cause elements to be incremented which will not help correct the final output at all. In fact, preliminary results from the computer simulations indicate that this is precisely what does occur with this first method.

In this way it is possible to hypothesize an incrementing procedure which does not make impossible demands on the complexity of individual elements and yet which can accomplish a rather sophisticated weight change.* It is a process that requires the rather large data

* It is of course unknown whether any analogous procedure occurs in biological neurons since even weight changes do not have any observed analog there. But the hypothesized fluctuating bias level suggests the somewhat uniform and periodic fluctuations observed in electrical potentials in a brain as given by an encephalograph.

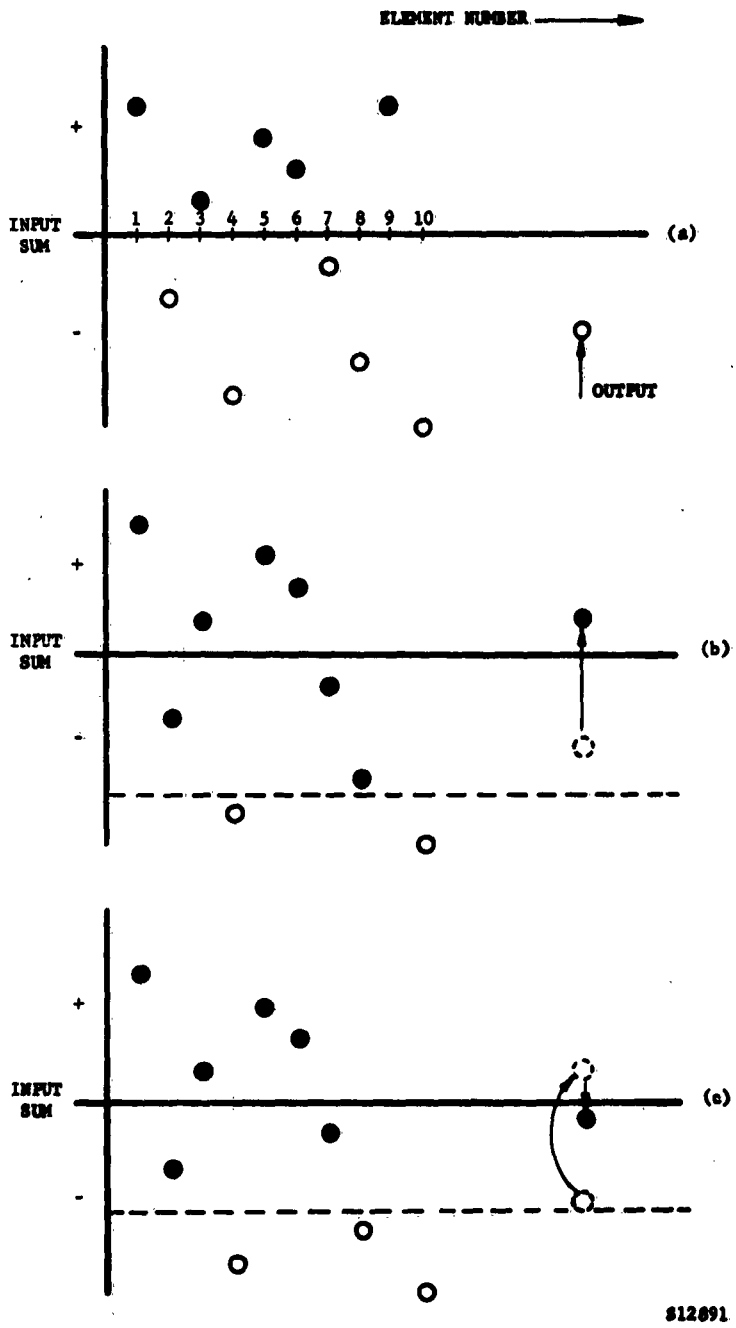
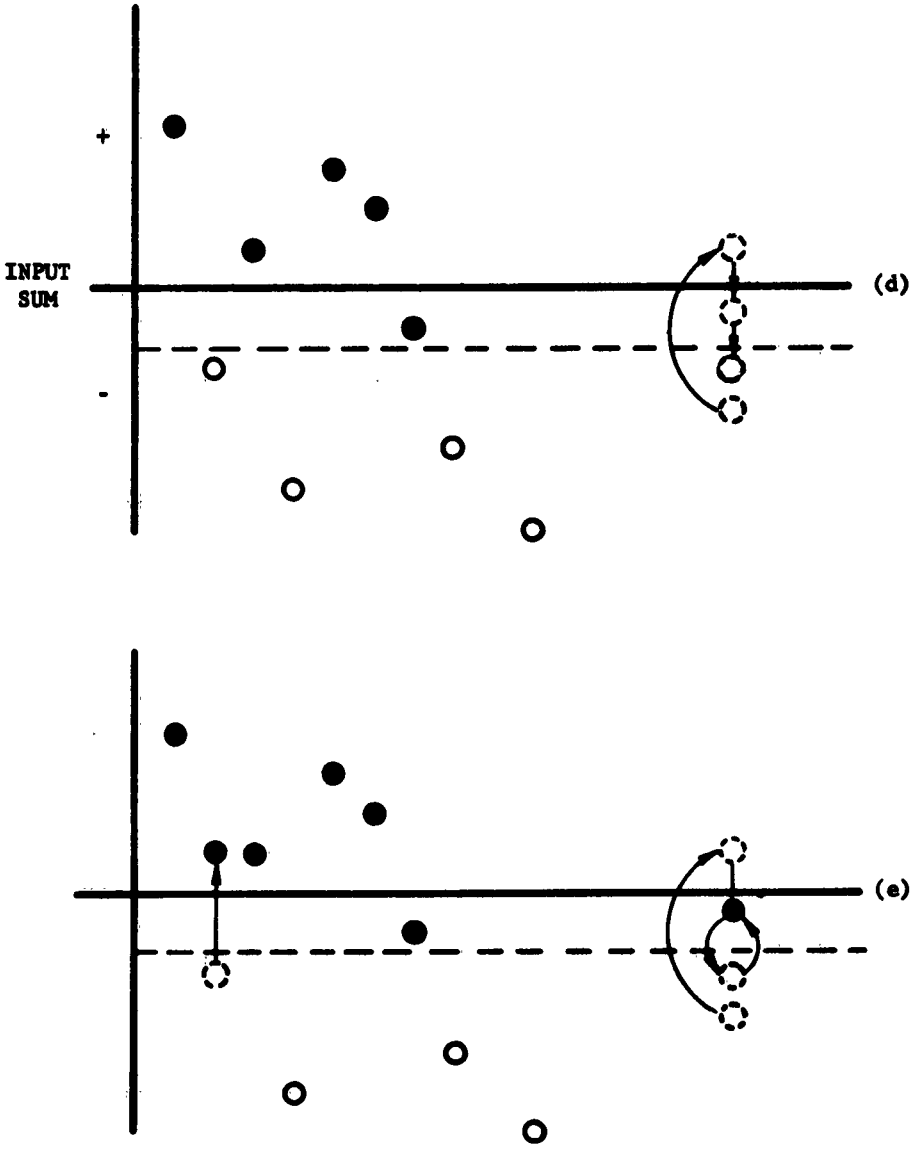


FIGURE 3.2. FALLING BIAS SELF EVALUATION



S12892

FIGURE 3.2. FALLING BIAS SELF EVALUATION (Continued)

processing capability of the network as it reverses various elements' outputs to see if they affect the final output value. But since this capacity is already required, little has been added to the network beyond the sum-dependence feature of each element's decision process.

A third principle which appears important here is that if a network is to be capable of learning a given logical function it should possess a sufficient complexity in terms of the total number of elements and of weights that this function can be realized by the network in a great multiplicity of ways. If one uses a network which is minimal in terms of numbers of elements and weights for the particular function, it is likely that there are relatively few essentially different combinations of weight values which will work. Consequently any learning process of the kind postulated is rather unlikely to stumble across one of these rare combinations.

To put this concept in other words conceive of each possible set of weight values throughout a given network as a point in a p-dimensional Euclidean space, p being the total number of weights in the network. Each different function which the network can realize can be associated with a certain p-dimensional volume in this space, namely the set of all points whose coordinate values (i.e., weight values) in the network cause this function to be generated. A function will tend to be easy, hard, or impossible to learn by the network in accordance with the relative size of volume it occupies in this space. But if the network has no redundancy with respect to a given function as regards the numbers of weights and elements, that function is likely to occupy a disproportionately small volume of the weight space, and hence be difficult or impossible to learn.

This situation is analogous to the case of the number of discrete levels of weight values necessary in a single linear threshold element. It can be shown that if each of the seven weights in a six-input threshold element can assume any of the 19 values, -9,-8,...,0,1, ...,9, then this element is capable of generating all possible six-input linear threshold functions.* But no known learning process will work for all such functions with weights so tightly restricted. Instead there is a great tendency for the element to generate only simple functions requiring low weights. In other words, in this case certain of the functions would occupy a disproportionately large share of all the 19^7 points in the weight space for the element.

* Appendix B, "A Magnetic Integrator for the Perceptron Program", Annual Summary Report, Aug. 1961, ASTIA #AD 264227.

There have been described in this section three important principles regarding the design of learning networks of linear threshold elements: 1) that it is essential to know in some way of the effect on the final output of an alteration on a given element in making the decision as to whether or not to make that alteration; 2) that changes made to weights must be made in such a way as to minimize the effect on the output values for other possible inputs which may previously have been learned; and 3) that a network should contain an excess of weights and elements over the minimum required to generate those functions which it is to be capable of learning. It was shown in the case of a single variable output how 1) could be implemented by restricting all interconnections to be positive and 2) by the use of a changing bias level furnished in common to all elements. In particular it was shown that although the decision-making involved in 2) required a considerable amount of data processing, practically all of this could be accomplished by using the already existing capacity of the network to generate outputs from inputs.

Computer simulations have brought to light a possible fourth principle which may be needed to avoid difficulties which have occurred. It has been observed that occasionally an initial set of weights would be selected for use with a logical function and network which would cause a small number of the network's elements to receive nearly all the weight increments, thereby greatly increasing the number of errors made during learning. Presumably there was in effect a much smaller network available for learning or weight change. This unfortunate circumstance arises whenever the small set of elements in question happens to always have small magnitude sums in comparison to that of other elements at the same time the output is in error, and when the logical complexity of the output in these erroneous cases exceeds the capabilities of this small number of elements.

A direct approach to this problem can assume that each element accumulates in some manner a "fatigue" factor which is a measure of the relative frequency with which an element has received weight increments. The greater is this factor the less susceptible is the element to incrementing. It might take the form of a variable weighting factor, f, on the sliding bias term, b, previously mentioned, thereby treating b very much like another input, though not a binary one. In any case, the need for this provision is still speculative pending further simulation results.

3.5 COMPUTER SIMULATION PROGRAM

Although a considerable amount of analysis is possible for the action of adaptable single linear threshold elements, no analysis is known

for the learning process of a network of them. It is our opinion that the best that can be hoped for in the future in the way of analysis of large networks is a statistical theory analogous, say, to the statistical mechanics theory in physics. Though efforts have been and will be made in this direction, for the present all learning theories will have to be evaluated by empirical means.

Until recent months simulations by hand or on the magnetic integrator learning machine were adequate for this purpose for the simple reason that each of the earlier postulated network models was shown to fail even for some very simple functions. However, hand simulations carried out more recently with the properties given in the last section had very good success, and for this reason a program was devised to more thoroughly test these models on an IBM 7090 general purpose computer. Some very good results have already been obtained and will be described in Section 3.6. This section briefly outlines the features of the program as it is now and with revisions which are now being planned.

The present program allows for the simulation of any network of up to 36 linear threshold elements and 35 binary input variables, the interconnections being specified by an interconnection matrix. The program is provided with a compiling routine which constructs a "straight line" program for computing sums and outputs from inputs which corresponds to whatever interconnection matrix may be assigned. This greatly increases the efficiency of the computation.

Provision is made for three types of incrementing rules. Mode 1 type of incrementing corresponds directly to a method described in Section 3.4. Whenever the network routine receives an input and then produces the wrong output, all sum values are scanned in elements whose output doesn't agree with that desired for the final element. The element whose sum is closest to zero is chosen to have its weights incremented. Increments of a fixed magnitude are applied to its weights in accordance with the rules laid down in Section 3.2 with the exception that no interconnecting weight (i.e., a weight applied to the output of an element when it serves as an input to another) is allowed to drop below some previously specified positive lower bound. The network is then re-evaluated using the new weights, and if the final output is correct, the next input combination is brought in. If it is still not correct this process is repeated with the same input combination until a correct answer is obtained.

A second mode of incrementing is provided which performs the mode 1 operation together with applying a smaller size increment to another

randomly selected element whose output doesn't agree with the desired final output. That is, mode 2 is similar to mode 1 except that two elements are incremented, one chosen as in mode 1, and the second chosen according to the sign of its sum but otherwise randomly. The purpose of this type was to break up what might otherwise be endlessly repeating cycles which could occur in mode 1. In simulation runs so far this mode has demonstrated its ability to do just that. However, care must apparently be used in choosing the size of the smaller increment lest it also break up the main learning process in what would otherwise be successful runs.

A third mode is provided which is similar to mode 2 except that every element whose output is in disagreement with the desired final output is to receive the smaller increment. The element to receive the large increment is selected as in mode 1. In particular if the large increment is set to zero, one obtains in mode 3 one of the previous methods of incrementing which was unsuccessful. Hence as may be expected, the small increment in mode 3 must be considerably smaller than the larger increment to avoid a slow learning rate.

The 2^{nd} and 3^{rd} modes were developed before the concepts of a sliding bias and a fatigue factor were formulated. It is expected that the latter will make the use of these modes unnecessary when the program is revised.

In these methods of incrementing the program allows for the use of different sized increments for interconnecting weights than for weights on the original inputs if desired.

Any desired logical function may be specified to be learned, and any set of initial weight values can be used with the network. The program then proceeds to run through the logical function, one input combination at a time. Whenever an error occurs, the weights are altered as mentioned above until the proper output occurs. When all input combinations have been presented if any errors have occurred, this process is repeated starting again with the first input. When all inputs can be consecutively presented with errors, the function is then learned and the program is caused to print out appropriate data such as the total number of errors made during learning, the total number of passes through the function, the final set of weights, the number of increments received by each element, etc. A maximum is set on the number of passes through all input combinations which, if reached, causes the program to cease the learning process and print the above data together with an indication that the network failed to learn the function in this case.

It is possible to specify that some selected portion of the network's element will not be subjected to weight change so that these portions of the network remain fixed during the learning process. The program also provides for switching to different types of incrementing during a learning process after a specified number of passes through the input combinations.

It is possible to put in a fixed bias term in the formation of sums for the elements which always works against the element's output agreeing with the final desired output. This results in forcing the weights to be so incremented that after the learning process is over and the desired function is learned, if this bias is now removed, then for each learned input-output case the elements whose outputs are essential to the correct final output have sum values different from zero by at least the value of the bias term. This simulates a process that undoubtedly would be required in any actual learning network to avoid having the learning process stop while some essential sum values were still marginal.

It has become increasingly clear as results of simulations have begun to accumulate that the efficiency of the learning process is critically dependent on the care with which the elements are selected for incrementing. When they are selected at random without regard to the size of the sum value (mode 2 without a large increment), or all of the proper sign are incremented equally (mode 3 without a large increment), then the learning process becomes chaotic and unlikely to succeed at all. When the element is chosen with regard to its sum value as in mode 1, a much more efficient process occurs. Nevertheless it is now apparent from the results obtained that even mode 1 is suffering from the disadvantage pointed out in Section 3.4, namely that often elements are incremented even though changing their output happens at the time to contribute little to the correctness of the final output. It is apparent that the "sliding bias" method outlined in that section deserves a chance to correct this deficiency. Hence a mode 4 type of incrementing is planned for the program revision now taking place which uses this sliding bias technique for selecting elements to be incremented.

It is also planned to allow for the computation of the "fatigue" factor mentioned earlier so that increments will be more evenly distributed throughout the elements than has occurred in several of the simulations.

The revised program will also provide for more than 36 elements and 35 input variables. This will allow the program to be used in the future for some moderately complicated tasks on, say, learning certain pattern recognition tasks. One can go in this direction only so far, however, before the cost of simulating a large learning network on a serial

computer becomes excessive. It requires about 7 microseconds on the IBM 7090 to add in each term (a weight times its binary input) in accumulating the sum value for an element. So if the network has a total of p weights, if the function to be learned has q input combinations, and if n repeats through all the input combinations are required for learning, then a total of about 10pqn to 20pqn microseconds are required for each learning experiment (allowing for repeated use of the network when there are errors.) It would not be difficult for this figure to become excessive for complicated networks and functions.

3.6 SIMULATION RESULTS

Since the program described in Section 3.5 has been in use for only a short time, the results are fragmentary. Yet they already demonstrate that substantial progress has been made toward designing successful learning networks, and that the principles discussed in 3.4 are essential to this design. Sections 3.6.a and 3.6.b give a brief description of these results.

a. Four-Input Functions

A total of 516 learning experiments were run on various four-input Boolean functions. As is shown in the appendix these four-input functions may be grouped into 83 symmetry types as far as their generation by linear threshold elements is concerned, and a representative from each of these 83 types was used in these trials. A variety of incrementing methods, three different sets of initial weight values, and two kinds of networks were used. Of the 516 trials, 4 of them had not been learned after 100 runs and were stopped at that time. The remaining 512 were learned with averages of 7 runs and 30 errors (a run being one pass through all sixteen input combinations, and an error being each occurrence of an improper output for a given input). These trials will be briefly described.

One experiment made a comparison of the effect of different numbers of elements in a network. The two networks had 6 and 8 elements, with the last element in each being permanently set at a majority function of the remaining 5 or 7, respectively. The other elements received only x-inputs, no interconnecting weights being used in this case. Two sets of 27 trials each were run which were identical in every way except for the size of the network used, and the fact that extra random initial weights had to be provided for the extra two elements in the 8-element network. A representative from each of 27 of the above 83 types was used including many from the difficult end of the list (e.g., the parity function). Three

different randomly chosen sets of initial weights were used. The incrementing method used mode 1 only (i.e., only one element incremented at a time and chosen according to its sum value). Letting r_{aver} be the average number of runs, r_{max} the maximum number of runs, e_{aver} the average number of errors, and e_{max} the maximum number of errors, the results of the two series may be summarized as follows:

	r_{aver}	r_{max}	e_{aver}	e_{max}
8-element network	3.2	4	11.7	20
6-element network	4.3	12	15.6	51

The larger network clearly has the advantage, particularly for the worst cases.

Next, a series of 249 trials was made, 3 for each of the 83 types, using the above three sets of random weights and the above 6-element network. The method of incrementing was: 9 runs on mode 1 followed on each tenth run by mode 2, with the small increment being one-fifth the size of the larger. The reason for this method of incrementing was to avoid endlessly repeating cycles by introducing an occasional random choice in the weight change process. Events proved that this was justifiable, and that in fact the random aspect was far too weak. The results were as follows:

	r_{aver}	r_{max}	e_{aver}	e_{max}
235 cases with $r \leq 20$	4.1	15	15.6	59
11 cases with $100 \geq r > 20$	44.6	84	182.6	341
3 cases with $r > 100$	100.	100	467	515

The last 3 cases listed were stopped at r = 100 and thus represent 3 of the 4 failures-to-learn mentioned before. As these data indicate there was a peculiar distribution of values for r and e. Actually only 17 cases altogether went beyond r = 7, which is encouragingly low. But of those that did, every case was apparently suffering from a repetitious cycle of incrementing which led in most of these cases to very high values for r and e, and was accompanied by exceedingly uneven distribution of increments made on the five elements with adjustable weights. For example, one case which reached r = 100 had made 4, 11, 7, 6, and 400 large size increments on the respective five elements. Obviously element number 5 was overworked.

In a third experiment the 14 difficult cases of the previous paragraph were each rerun three times, everything being left unchanged save for the use of three different methods of incrementing. The first method changed only the size of the small increment used by mode 2 during each tenth run. It was made equal in magnitude to the large increment, rather than one-fifth size. The second method used the mode 2 every time rather than every tenth time, the small increment being a fifth of the large one as before. The third method introduced still more drastic variation by using mode 1 as before for the first 10 runs, and then for the next 10 runs using mode 2 with no large increment, only a small one equal to the increment used in the first 10 runs. In other words, for 10 runs the choice of element to increment was based upon sum value, but for the next 10 runs it was by random choice. The third 10 runs reverted again to mode 1. Of course, in the first and third methods all runs exceeded 9 since there was no change over the previous trials for that period. The results were as follows:

	r_{aver}	r_{max}	e_{aver}	e_{max}
1^{st} method	20.0	34	90.1	156
2^{nd} method	8.6	15	38.9	76
3^{rd} method	22.4	30	127.1	220

There is a clear improvement over the previous runs for the first and third methods since none of the 232 runs for which $r \leq 9$ could be adversely affected. (Conceivably the three cases with r in the teens might be adversely affected but this is unlikely.) In other words, what was lacking previously was a method of jolting a network out of a repeating cycle. Method 3 appears to have gone too far in that direction by comparison to method 1. (However, remarkably enough, 4 of the 14 cases in method 3 were terminated when no elements were being chosen for incrementing according to their sum value, only randomly.) Method 2, which appears much better than the others, would require further testing to justify that conclusion. Since it changes the incrementing method from the very beginning it would have to be tested for all the other 235 cases to make an exact comparison. The next experiment indicates that in fact this second method is a good one.

The fourth experiment used seven different methods of incrementing. For each method, the last 11 functions of the listed 83 were each used 3 times starting with the previous 3 sets of initial weights. This made a total of 231 trials. The 6-element network was again used. The first three methods of incrementing used mode 2 exclusively with the

small increment being 1/10, 1/5 and 1/2 of the size of the large increment for the first, second, and third series, respectively. Mode 3 was used for the next three (i.e., all elements with a wrong output receive a small increment whenever the final element is in error) with the ratios of small to large increment being 1/50, 1/20, 1/10 for the fourth, fifth, and sixth series, respectively. The seventh series (which is actually a part of the group of 249 trials and is listed here for comparison) used mode 1 except that every tenth run mode 2 occurred as before. The results were:

	r_{aver}	r_{max}	e_{aver}	e_{max}
1^{st} series, mode 2, S/L = 1/10	7.3	59	31.5	285
2^{nd} series, mode 2, S/L = 1/5	5.4	17	23.6	88
3^{rd} series, mode 2, S/L = 1/2	5.8	12	25.5	43
4^{th} series, mode 3, S/L = 1/50	12.9	100	55.7	443
5^{th} series, mode 3, S/L = 1/20	6.1	33	28.9	136
6^{th} series, mode 3, S/L = 1/10	6.0	20	26.2	81
7^{th} series, mode 1	12.1	100	52.9	515

Note that two failures to learn occurred here, one in the last series, which was one of the previous three mentioned, and a second in the fourth series with S/L = 1/50. These results again show a clear trend for improvement on the mode 1 method by the addition of other increments as in modes 2 and 3. A comparison of the second and third series indicates that by making the small increment sufficiently large, the average learning time begins to get larger even though the worst cases are better.

b. Six-Input Functions

A second group of 180 learning experiments was made on four different six-input Boolean functions. These functions will be denoted by A, B, C, and D. Function A could actually be obtained from a single element using the weights 8, 7, 6, 5, 4, 3, 2 but it can be shown that this would require something of the order of 1000 errors before being learned. Functions B and C were randomly selected, while D was the six-input parity function (i.e., its output is 1 when an odd number of its inputs are). A complete set of 64 input combinations was used for each.

In each of the series of experiments to be described there was provided only one set of initial weights, which was used only by function

A on its first trial. Thereafter the four functions appeared in sequence: A, B, C, D, A, B, etc. with each function using as its initial weight values the final set of weights of its predecessor. Since a set of weights appropriate to one of these functions would give wrong answers for the succeeding function, this procedure approximated the use of random initial weights for each trial.

In the first series of 56 trials a 22-element network was used with the final element a fixed majority function of the other 21. There were no other interconnecting weights so that learning occurred only on one layer as in the case of the networks used for the four-input functions. Each of the 21 elements received each of the six input variables and had a variable threshold weight, making a total of 147 variable weights. The incrementing method was to use mode 1 for 9 runs followed each tenth run by mode 2 with S/L = 1. The initial weights were all zeros, except for the fixed weights in the final element. The values of r, the total number of runs, and e, the total number of errors, are given in the following table for each trial.

	A		B		C		D	
	r	e	r	e	r	e	r	e
1	10	55	15	302	16	208	5	75
2	5	52	18	215	9	97	4	51
3	7	47	18	192	8	110	4	48
4	3	40	7	69	10	98	6	68
5	4	43	7	82	4	47	6	46
6	5	46	5	70	7	84	5	54
7	4	45	4	44	7	78	7	64
8	3	32	4	53	7	74	3	40
9	4	42	6	51	4	60	4	46
10	5	35	5	54	9	67	4	39
11	4	44	6	53	6	70	5	53
12	4	40	5	54	7	68	4	36
13	4	37	4	48	6	46	3	35
14	4	36	5	52	7	67	10	73

The second series of 24 trials was just like that of the preceding series except that the initial weights were randomly selected for use in the first trial by function A. As before each succeeding trial used its predecessor's weight values. The results were as follows:

	A		B		C		D	
	r	e	r	e	r	e	r	e
1	3	29	7	76	5	78	6	70
2	4	37	5	70	7	71	7	81
3	4	44	5	49	5	57	5	37
4	3	39	5	59	6	64	4	42
5	5	36	5	60	5	65	5	51
6	5	44	6	71	3	43	4	37

The results of these two series strongly indicate that to give a large network such as this one all zero weights initially is to severely handicap it. It apparently required about 10 or 15 successive trials in the first series to sufficiently "spread out" the activities of the 21 first layer elements to overcome this initial disadvantage. The second series with random initial weights showed no such difficulty. If the first 16 trials of the first series are ignored, the averages for r and e for the remainder of the two series are:

A		B		C		D	
r_{aver}	e_{aver}	r_{aver}	e_{aver}	r_{aver}	e_{aver}	r_{aver}	e_{aver}
4.1	39	5.3	59	6.0	65	5.1	50

In the third and fourth series, variable interconnecting weights were used for the first time. The network consisted of 22 elements arranged in three layers, 10 elements in the first layer, 11 in the second, and the final element in the third. The final element was a fixed majority function of the 11 in the second layer. Each element in the second layer received the outputs of each of the 10 elements of the first layer with weights which were variable. As before each of the 21 first and second layer elements received all six input variables. Thus there was a total of 257 variable weights. The third series of 20 trials used only mode 1 with x-weight and y-weight increments being equal. The y-weights were not allowed to go below the magnitude of one such increment. The fourth series of 20 trials differed from the third only in that 3 runs of incrementing only on the first layer alternated with 3 runs of incrementing only on the second layer, rather than incrementing being permitted on both layers at

the same time as in the third series. Both series were started at zero initial weights for the x-weights and at the lowest allowed values for the y-weights. The results were as follows:

Third Series:

A		B		C		D	
r	e	r	e	r	e	r	e
20	126	25	348	10	151	6	95
6	47	11	165	6	70	5	53
5	54	8	100	11	101	4	52
4	37	9	85	4	60	5	62
4	44	6	72	9	85	6	56

Fourth Series:

A		B		C		D	
r	e	r	e	r	e	r	e
10	61	22	382	16	297	14	160
5	42	13	128	11	132	9	104
6	49	11	144	9	140	8	92
6	55	12	189	8	95	10	119
5	46	11	125	10	115	14	202

The following gives the averages of the first 20 trials of the first series, of the third series, and of the fourth, for purposes of comparison since each was started with zero initial weights:

	A		B		C		D	
	r_{aver}	e_{aver}	r_{aver}	e_{aver}	r_{aver}	e_{aver}	r_{aver}	e_{aver}
1^{st} series	5.8	47	13.0	152	9.4	92	5.0	58
3^{rd} series	7.8	62	11.8	154	8.0	93	5.2	64
4^{th} series	6.4	51	13.8	194	10.8	156	11.0	135

These results indicate that incrementing on one layer at a time in the fourth series gave a somewhat poorer performance than incrementing with both layers at the same time. It is surprising to note that the

addition of 110 y-weights in the third and fourth series produced little, if any, improvement over the first 20 trials of the first series. This matter receives further attention in the next three series.

The final three series of trials was designed to further test the performance of networks with variable y-weights. Each used mode 1 for 9 runs followed on each tenth run by mode 2, and each used a network of 12 elements with the final element being a fixed majority function of the other 11. Each of the 11 received all x-inputs. The series differed in that the fifth had no y-weights, the sixth had 55 y-weights (all that could be added), but these were fixed, and the seventh had the same network as the sixth but with variable y-weights (except those of the final element). Thus the seventh series was an example of learning on 11 layers simultaneously. The reduction to 12 elements was intended to compare further the effect on learning rates of varying the size of a network. That there was such an effect is clear from the fact that of the 60 trials, 15 had failed to succeed in learning the required function after r = 25 (at which time the program stopped them). The failures are indicated by asterisks in the tables below. Each series was initiated by the same set of randomly selected x-weights. The sixth series had a y-weight fixed at five times the size of a mode 1 weight increment. The seventh started initially with these same y-values. The results were as follows:

Fifth Series:

	A		B		C		D	
	r	e	r	e	r	e	r	e
1	4	39	25*	405*	25*	393*	5	62
2	6	50	25	240	17	194	6	72
3	5	43	10	114	14	153	7	78
4	6	57	25	274	16	192	6	70
5	4	35	24	333	10	146	15	129
averages	5.0	45	21.8	273	16.4	216	7.8	82

Sixth Series:

	A		B		C		D	
	r	e	r	e	r	e	r	e
1	6	56	25*	366*	24	250	23	153
2	6	48	6	114	25*	377*	15	141
3	4	37	14	221	25*	503*	25*	243*
4	4	50	25*	549*	20	283	17	217
5	6	53	25*	369*	25	372	25*	452*
averages	5.2	49	19.0	324	23.8	357	21.0	241

Seventh Series:

	A		B		C		D	
	r	e	r	e	r	e	r	e
1	5	44	19	277	12	177	7	72
2	7	56	25*	374*	10	113	15	159
3	5	65	25*	377*	25*	315*	25*	529*
4	7	55	25*	346*	16	314	25*	345*
5	5	37	18	215	19	234	5	52
averages	5.8	51	22.4	318	16.4	231	15.4	231

Again the networks with y-weights added failed to live up to expectations, and in these trials actually acted as a handicap, particularly in the sixth series when they were fixed. It was observed in both the sixth and seventh series that the elements with the larger number of inputs received much the smaller share of increments. This is due no doubt, to the tendency for their sums to be larger, thereby making it difficult to be chosen for incrementing under the rules of mode 1. It is felt, however, that the difficulty lies deeper than this in view of the results of the fourth series where the second layer was given an equal opportunity to the first for receiving increments.

At the present time the most likely explanation lies in the previously mentioned deficiency in mode 1 type incrementing when applied to multi-layer networks -- namely, that it can select elements for incrementing, which may not be important at the moment in helping to correct

the final output. This cannot happen in the networks used which had no y-weights and all elements led directly to the final element in a fixed majority function. (It could, of course, if the final element had used a different fixed function with unequal weights.) In any case the planned future use of another mode of incrementing (as explained in previous sections) which tests the effect of changing the output of an element in a network before incrementing its weights, should settle this question.

Another interesting and encouraging observation that can be made about the last three series' results is that while the performance was considerably worse in the 12-element networks over the 22-element ones for functions B, C, and D, this was not true for A. Function A, which can be generated (with difficulty) by one element, was learned about equally well by either network. This shows how a function which can be generated by a small fixed network, is likely to require only a small network to be learned with reasonably few repeated errors.

As is known, for n binary inputs there are 2^{2^n} possible Boolean functions of them, and a network which is sufficiently flexible to learn any of them must perforce have a proportional flexibility in terms of the number of its variable weights. Hence, for large values of n it is clearly out of the question to envision networks large enough to realize or learn all possible functions of the inputs. However, if it remains true that the size of a network required to learn a function is roughly proportional to the minimum size of fixed network required to generate the function, this would still make it useful to deal with large numbers of inputs in learning networks. The functions one is likely to want to have learned will presimably also be likely to be capable of generation, and therefore of learning, by relatively small networks (i.e., small when compared to that needed to learn all 2^{2^n}.)

the final output. This cannot happen in the networks used which had no y-weights and all elements led directly to the final element in a fixed majority function. (It could, of course, if the final element had used a different fixed function with unequal weights.) In any case the planned future use of another mode of incrementing (as explained in previous sections) which tests the effect of changing the output of an element in a network before incrementing its weights, should settle this question.

Another interesting and encouraging observation that can be made about the last three series' results is that while the performance was considerably worse in the 12-element networks over the 22-element ones for functions B, C, and D, this was not true for A. Function A, which can be generated (with difficulty) by one element, was learned about equally well by either network. This shows how a function which can be generated by a small fixed network, is likely to require only a small network to be learned with reasonably few repeated errors.

As is known, for n binary inputs there are 2^{2^n} possible Boolean functions of them, and a network which is sufficiently flexible to learn any of them must perforce have a proportional flexibility in terms of the number of its variable weights. Hence, for large values of n it is clearly out of the question to envision networks large enough to realize or learn all possible functions of the inputs. However, if it remains true that the size of a network required to learn a function is roughly proportional to the minimum size of fixed network required to generate the function, this would still make it useful to deal with large numbers of inputs in learning networks. The functions one is likely to want to have learned will presimably also be likely to be capable of generation, and therefore of learning, by relatively small networks (i.e., small when compared to that needed to learn all 2^{2^n}.)

SECTION 4

PARALLEL LEARNING NETWORKS

The motivation for considering parallel networks of learning elements stems from two sources. One is that a considerable amount of data of interest for learning processes occurs in image or two-dimensional form. Another is that only networks consisting of rather large numbers of elements can hope to cope with the useful but difficult functions associated with practical learning tasks.

This section describes a technique for implementing large two-dimensional iterative arrays of learning elements, and presents a simple example. The technique has also been used to implement large fixed-logic (non-learning) networks, and has been applied to potentially practical pattern-recognition processes. This work has been previously reported,* and more details of the applications to fixed-logic recognition networks may be obtained from the reference.

4.1 PRINCIPLES OF MECHANIZATION

The mechanization to be described applies to arrays of identical single-level logic decision elements in two dimensions. The technique can obviously be extended to multi-level logic by employing the output (array) of one stage as input to subsequent stages. This refinement has been performed on a fixed-logic basis. No attempt has yet been made, however, to reduce the multi-stage learning procedures described in the previous section to this particular form.

---

*J. K. Hawkins, C. J. Munsey, "A Natural Image Computer", Optical Processing of Information, Spartan Books. (to be published).

The technique employed to mechanize a two-dimensional array of single-stage logic elements, each operating on a local region of an input array or image, involves two steps. The first step is an optical system similar to that used in performing two-dimensional cross correlation. The system is illustrated in Figure 4.1. The two planes, T_1 and T_2, are transparencies. T_1 corresponds to a set of input variables (x's). T_2 contains regions whose transmittance is proportional to the coefficients (g's) required to mechanize a given logical function. It can be seen that with a diffuse, incoherent light source, I, the quantity of light energy, e, reaching any point on the resultant plane is

$$\frac{e}{I} = x_1 g_1 + x_2 g_2 + \ldots + x_N g_N. \tag{1}$$

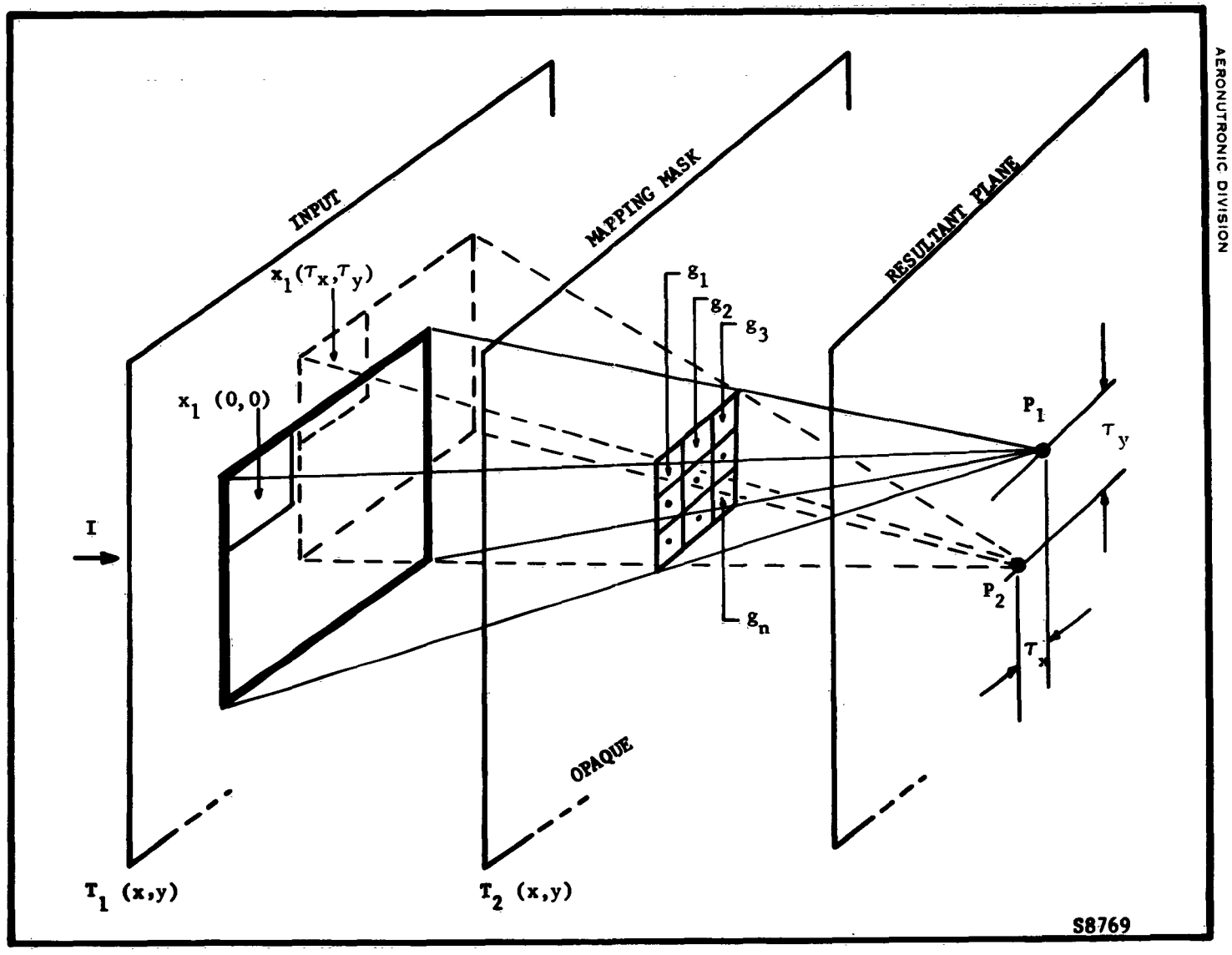
This is the direct analog of the weight-summing function of linear-input learning elements discussed in Section 3. In particular, the input sum, s, is given by

$$s = \sum_{i=0}^{n} c_i x_i \tag{2}$$

Note further that the quantity in (1) -- with the same coefficients -- is formed simultaneously with respect to every region containing input variables on the input field. Thus the quantity in (1) is formed at point p_1 with respect to the input variables within the heavy border. At p_2, the same quantity is formed with respect to variables within the dotted border, and so on, everywhere on the resultant plane. No physical motion is involved. All results are formed simultaneously and in parallel. The effect is entirely equivalent to that of wiring a field of Kirchhoff summing networks.

Because optical transmittance, and electrical conductance, are constrained to be positive, some technique must be employed to obtain the effect of negative coefficients in (2). Negative coefficients are required for most useful functions. The effect of a negative coefficient can be obtained if the complements, $\bar{x}_i$, of the input variables, x_i, are available. In a system in which variables can only be 0 or 1 (no light energy, or light energy) a simple manipulation converts (1) into the appropriate form.

In order to complete the requirements for a logical function, it is now necessary to carry out a decision operation on the resultant plane of Figure 4.1. This corresponds to the "sgn" operator of the previous section. Any light-amplifying device with sufficient amplification is



Ford Motor Company
AERONUTRONIC DIVISION

FIGURE 4.1. ILLUSTRATIVE CPTICAL SUMMATION SYSTEM

suitable, in principle, for this purpose. The only requirement is that each cell of the amplifying array be individually capable of sensing a light input, and producing a light output if the input exceeds a threshold value. The last requirement is necessary if the output of one stage of the computing process is to serve as the input to a subsequent stage.

An example of a planar decision medium is photographic film, which has been employed as the light amplifying medium in a number of experiments. Qualitatively, photographic film may be regarded as a light-amplifier if we consider the sequence: a) expose film to input light L_i, b) develop, and c) project light source through resulting transparency to produce light output, L_0. The ratio L_0/L_i can be extremely large (consider, for example, a movie projector), and therefore power amplification is obtained. In effect, the "contrast" behavior of ordinary film is employed to perform a decision operation.

4.2 PARALLEL LEARNING

The above process can be regarded, in one sense, as a very large number of experiments in parallel. That is, each small region of the input array may be regarded as containing a sample configuration of the local input variables. Some configurations are to be classified as 1, others as 0. Thus one can imagine presenting all 2^n possible configurations of n (local) input variables simultaneously on one parallel array. In practical imagery, of course, only a small fraction of the 2^n possible input configurations will actually occur.

For a learning process involving a specified desired output for each input configuration, we can also imagine that all of the specified desired output values are presented simultaneously. The desired output may therefore be regarded as a two-dimensional field of binary 1's and 0's, located in 1:1 geometrical correspondence to the presence or absence of the appropriate input example. The actual network output also appears in 1:1 geometrical correspondence to the local input patterns.

The rule for changing coefficients in a single-stage decision network to guarantee convergence to a solution, if it exists, can be represented by

$$\Delta g_i = x_{ij}(D_j - A_j) \qquad (3)$$

in which

x_{ij} = i^{th} variable of j^{th} configuration of input variables,

D_j = desired network output for j^{th} configuration, and

A_j = actual network output for j^{th} configuration.

This formula implies a sequential presentation of input configurations, $j = 1,2,\ldots,m$. When all local regions of a two-dimensional field are regarded as individual configurations, then we can state the rule for changing the coefficients in the following form. At each step (presentation of an input field),

$$\Delta g_i = \sum_j x_{ij}(D_j - A_j) \qquad (4)$$

in which the subscript, j, represents position (in two dimensions) on the field.

The crucial point to note is that (4) has precisely the same form as (1) or (2), and can thereby be mechanized by the same optical system as previously discussed. Note that at least two steps are involved in the learning process. First, the coefficient values, g, are placed at the position of the mapping mask of Figure 4.1, and the actual output field, A_j, is produced by performing a decision operation on the resultant light sum. The difference field, D-A, is produced by noting that for binary decision levels it is equivalent to the "exclusive-or" function, $D \oplus A$, and can be generated in the form $D\overline{A} + \overline{D}A$. Finally, Δg must be formed according to (4). This is accomplished by placing (D-A) at the position of the mapping mask of Figure 4.1.

The proof of convergence for this process is obtained by drawing a correspondence to existing proofs. In one proof,* the input patterns are regarded as vectors, $\overline{X}$, in n-dimensional space, in which n is the number of variables per pattern. The effect of (3) is to cause the coefficient vector, $\overline{G}$, to move a unit distance in the direction of $\overline{X}$, if the actual output does not agree with the desired. Virtually any incrementing scheme which has this effect can be shown to converge. In the case of (4), however, the movement of $\overline{G}$ in the direction $\overline{X}$ is no longer a unit distance, but is dependent upon the number of times the particular configuration $\overline{X}$,

*Appendix A, "A Magnetic Integrator for the Perceptron Program", Annual Summary Report, Aug. 1961, Aeronutronic Div., Ford Motor Co., ASTIA #AD 264227.

appears on the input image. We may therefore regard the model as one in which the input vectors have different lengths, and the movement of $\overline{G}$ is proportional to that length. Diagrammatically, the two models are represented in Figure 4.2. Convergence can nevertheless be demonstrated by noting first that $|\overline{G}|$ is still bounded (although the bound may be much larger than in the sequential case if a large number of local patterns on the input field are identical). Secondly, the component of $\overline{G}$ along the solution (if it exists) is always positive. Thus $\overline{G}$ must eventually reach a position sufficiently close to the solution, such that D = A everywhere.

In order to illustrate this approach, a known linearly-separable function was tested by hand simulation. The input field and desired output field are presented in Figure 4.3. The function corresponds to $D = \overline{x}_3(x_2+x_1)$, with the variables arranged as shown in Figure 4.3. The results of performing the operations specified previously are illustrated in Figure 4.4. The desired output is reproduced for comparison. The initial values of the coefficients were taken to be zero, so that the first value of the actual output field, $A_i(t=0)$, is simply zero everywhere. Successive output fields are subscripted in Figure 4.4 in sequence, the last A_7, agreeing everywhere with the desired field and thereby terminating the process, by (4).

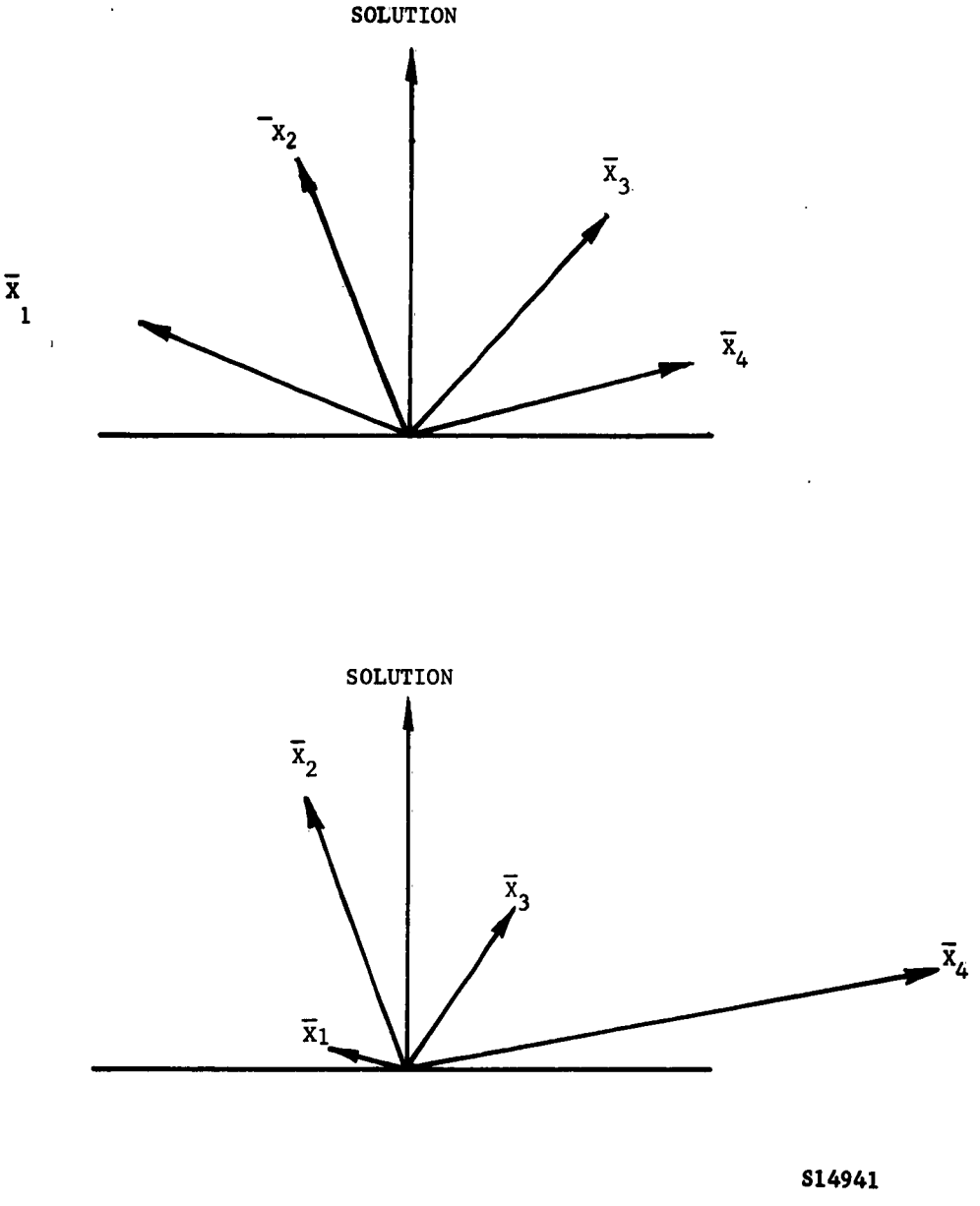
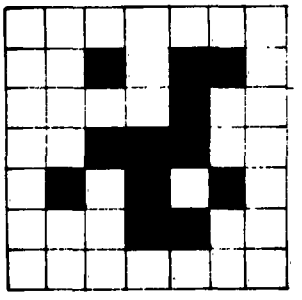
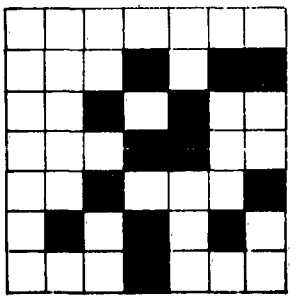


FIGURE 4.2. REPRESENTATION OF INPUT PATTERNS



INPUT (x)



DESIRED (D)

X_3	X_1
X_2	D

VARIABLE ARRANGEMENT

S14942

FIGURE 4.3. FUNCTION FIELDS

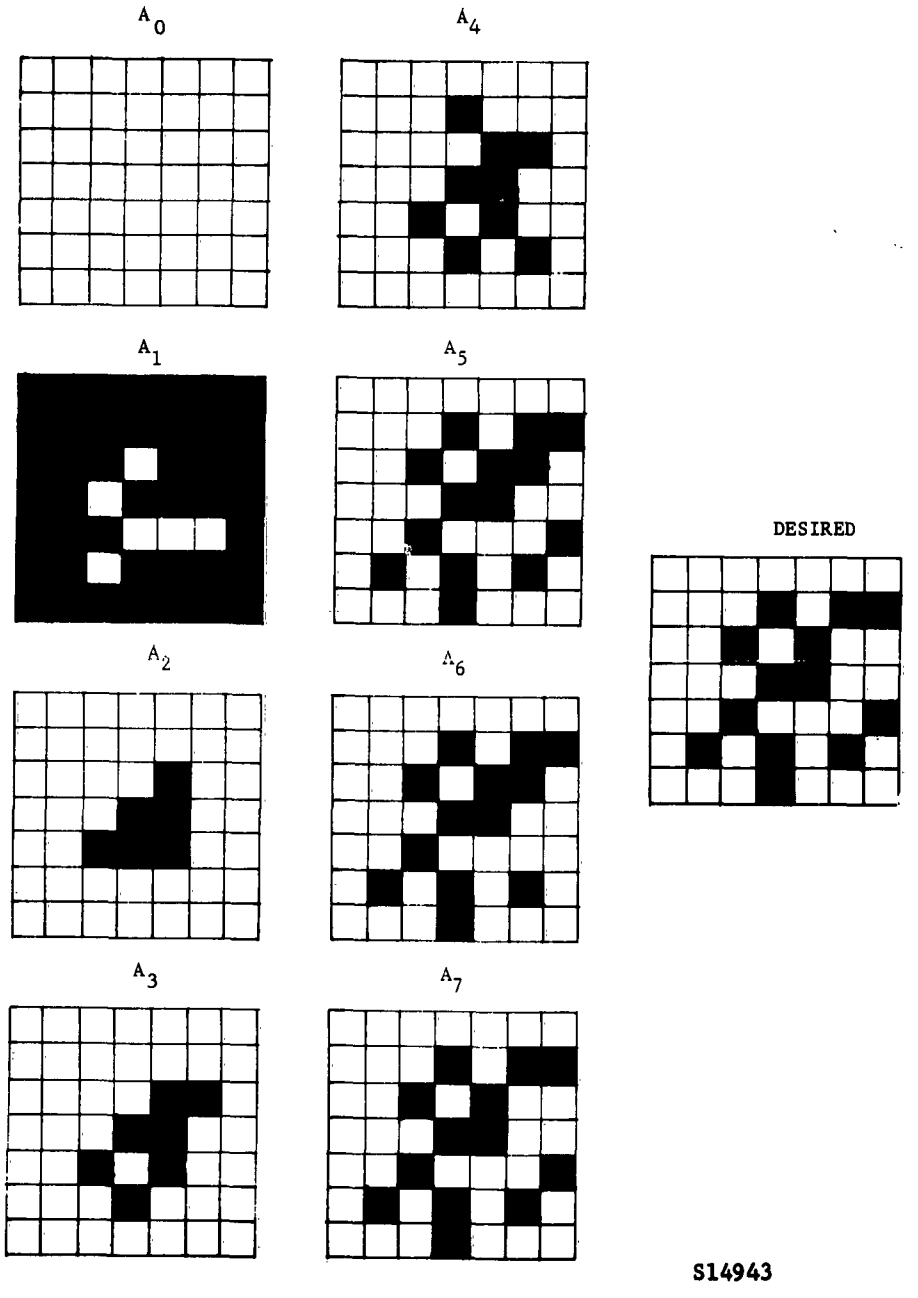


FIGURE 4.4. PARALLEL LEARNING EXAMPLE

APPENDIX

TABLES OF GENERATION OF FOUR VARIABLE BOOLEAN FUNCTIONS USING THRESHOLD ELEMENTS

(Prepared by R. A. Stafford)

The accompanying tables give a compilation of means of generating Boolean functions of up to four variables using linear threshold elements. There are 2^{16} such Boolean functions, out of which 1882 require only a single linear threshold element, 50,284 require two such elements in cascade, while the reamining 13,370 require three elements (see figure 1).[1]

A linear threshold element with inputs $A_1, A_2, \ldots, A_n$ and output B is defined as one for which there are values or weights $w_o, w_1, \ldots, w_n$ such that $B = \text{sgn}(w_o + w_1A_1 + \ldots + w_nA_n)$ where $A_1, \ldots, A_n$ each can assume only the two values +1 and -1.[2] The constant term w_o will be referred to as the threshold weight.

For each Boolean function $f(A_1, \ldots, A_n)$ let there be defined an augmented function, f*,

$$f^*(A_o, A_1, \ldots, A_n) = A_o f(A_1, \ldots, A_n) + \bar{A}_o \overline{f(\bar{A}_1, \ldots, \bar{A}_n)},$$

involving an additional variable A_o (the symbols in this equation referring to logical operations.) Such an augmented function must always have the antisymmetric property

$$f^*(\bar{A}_o, \ldots, \bar{A}_n) = \overline{f^*(A_o, \ldots, A_n)}.$$

Conversely, every function with this antisymmetric property can be obtained by augmenting an appropriate function of one fewer variables. In particular if f is a linear threshold function, $f(A_1, \ldots, A_n) = \text{sgn}(w_o + w_1A_1 + \ldots + w_nA_n)$, then $f^*(A_o, A_1, \ldots, A_n) = \text{sgn}(w_oA_o + w_1A_1 + \ldots + w_nA_n)$. More generally, if $f(A_1, \ldots, A_n)$ is generated by a number of linear threshold

---

1. These figures agree with those given by R. C. Minnick in IRE Trans. on Electronic Computers, Vol. EC-10, March, 1961.

2. In the truth diagrams given in the tables, the +1 and -1 values are represented by the classical Boolean symbols, 1 and 0 respectively.

elements in cascade, then $f^*(A_o,A_1,\ldots,A_n)$ is obtained by regarding the threshold weight in each element as being multiplied by another variable A_o, rather than being constant.

This concept can be used to define a useful kind of symmetry relation among Boolean functions. For the purposes of this report the functions $f(A_1,\ldots,A_n)$ and $g(A_1,\ldots,A_n)$ are said to lie in the same symmetry class whenever their corresponding augmented functions $f^*(A_o,A_1,\ldots,A_n)$ and $g^*(A_o,A_1,\ldots,A_n)$ can be made identical by making an appropriate transformation on the variables, $A_o,A_1,\ldots,A_n$, of one of the functions. The transformation is to be some permutation (possibly the identity) on the variables combined possibly with a substitution of some or all of them by their respective complements. For example, $f(A_1,A_2,A_3) = A_1A_2 + \bar{A}_1A_3$ and $g(A_1,A_2,A_3) = \bar{A}_2(A_1\bar{A}_3 + \bar{A}_1A_3)$ belong to the same symmetry class in this sense because

$$f^*(A_o,A_1,A_2,A_3) = A_2(A_oA_1 + \bar{A}_o\bar{A}_1) + A_3(A_o\bar{A}_1 + \bar{A}_oA_1) = g^*(\bar{A}_3,A_o,\bar{A}_2,\bar{A}_1)$$

The usefulness of this type of classification lies in the following fact. If f and g bear this symmetric relation to each other and if f is generated by some particular arrangement of linear threshold elements, then g can also be generated by the same arrangement of elements and using the same sets of weights in each element except that the weights attached to the original inputs $A_1,\ldots,A_n$ and the threshold weight of each element are to be permuted and changed in sign in accordance with a transformation on $A_1,\ldots,A_n$ and A_o which renders g^* identical to f^*. For example, the f defined in the previous paragraph can be generated by two elements, $X = \text{sgn}(1 + A_1 + A_3)$ and $f(A_1,A_2,A_3) = Y = \text{sgn}(2X - 1 - A_1 + A_2)$, or in the more compact notation to be used hereafter,

	X	A_o	A_1	A_2	A_3
X		1	1	0	1
Y	2	-1	-1	1	0

.

Correspondingly, g is generated by

	X	A_o	A_1	A_2	A_3
X		-1	1	0	-1
Y	2	0	-1	-1	1

,

which represents a permutation and sign change on the set of weights for f

in accordance with the transformation described in the previous paragraph which rendered f* and g* identical.

Thus in a study of means of generating functions with linear threshold elements it suffices to select only one representative from each symmetry class. For four variables there are 83 such classes,[3] so the accompanying tables deal only with 83 different functions, each a representative of one of these classes. These are expressed in the augmented form as functions of the five variables, A, B, C, D, E. Of these the first 7 can be generated with one element, the next 62 with two elements, while the last 14 require three elements each.

The tables are principally devoted to enumerating, for each of the 62 two-element functions, all possible linear threshold functions which will serve as that in the first or auxiliary element, X (figure 1b). Moreover, these are grouped according to the way in which the corresponding regions in the five-dimensional weight space fit together.[4] That is, if a set of X-functions had the property that the set of all points $(W_A, W_B, W_C, W_D, W_E)$ which will yield one of these given functions in X forms a single connected region[5] then these functions were grouped together.

Table II gives a listing for each of the 62 two-element functions of weight values for each acceptable X-function except for those cases which can be derived from listed weights by an appropriate transformation as described in the later paragraphs. Also listed together with each of these is a set of weights (usually the smallest integral values that can be found) which will serve for the Y element. (In general, for a given X-function, a number of Y-functions are possible, but only one is listed.) The small letters indicate the groupings mentioned in the previous paragraph.

---

3. The number of such symmetry classes increases very rapidly as the number of variables increases, as indicated by the chart:

No. of variables	1	2	3	4	5
No. of classes	1	3	7	83	109,950

4. These tables were developed in connection with a study of constructing cascaded-element networks with variable weights which after a period of learning are to be capable of automatically assuming values which will render the output a certain desired function of the inputs.

5. More precisely, if weights can be varied continuously while the truth table changes one entry at a time while changing from one X-function to another, then these are grouped together.

Table I gives for each group listed in Table II a single truth diagram for the X-functions of the group. This diagram contains entries only where all these functions are in agreement. Also a truth diagram for Y-functions is given for each such group. It contains only entries that are common to all Y-functions which will generate the desired function when used with one of the X-functions of the group.[6]

If a transformation (permutation and complement change) of the variables A,B,C,D,E leaves the desired function unchanged, then any acceptable X-function (i.e., one which serves to generate the desired function) can be subjected to this transformation without impairing its acceptability. For economy of space, Tables I and II contain only one version of any set of X-functions which can be obtained, one from another, by such transformations leaving the desired function invariant. Thus, in the example for $f(A_1,A_2,A_3)$ given above, the transformation $(A_o,A_1,A_2,A_3)\rightarrow(A_1,A_o,A_3,A_2)$ leaves f invariant but produces a different X than that given. Two other X-functions could be obtained from such transformation on the given X-function. Of these four only one will appear in the tables.

A second saving in space is made by listing in the tables only X-functions which require a positive weight on their input to Y. (This weight and all those of X can always be reversed in sign without changing the output of Y.) However, the numbers in Table I indicating the number of acceptable X-functions possible for the desired function do include both kinds.

Table I also includes a single generation of each one-element or three-element function. No attempt is made to find all X,Y pairs possible for three-element functions due to their enormous number.

Table I is best explained by using the previous f again as an example. It is of the type of function number 15, $f(A,B,C,D,E) = A(CD + \overline{CD}) + B(\overline{CD} + C\overline{D})$. The truth diagram for this as indicated at

---

6. It turned out that in each of the 62 cases any linear threshold function which agreed with a partial X-truth diagram in Table I belonged to that corresponding group. Another fact observed was that the partial truth diagrams for Y in Table I could always be filled in with 0's on the right side (X=1) and 1's on the left side, and the resulting function would serve for all X-functions in that group.

the top of the page is arranged:

	0	0	1	1	D
	0	1	0	1	E
000	0	0	0	0	
001	0	0	0	0	
010	0	0	1	1	
011	1	1	0	0	

ABC

The other half for A = 1 can be deduced from the requirement $f(A,B,C,D,E) = \overline{f(A,B,C,D,E)}$.[7] There are 160 distinct linear threshold functions (the number in parenthesis above the truth diagram) which will serve for X. All the 20 threshold functions which agree with the partial diagram

	0	0	1	1	D
	0	1	0	1	E
000					
001					
010	0	0	1	1	
011	1	1	1	1	

ABC

will serve for X and their corresponding w-regions are connected. All suitable Y-functions for members of this group agree with the diagram:

	0	0	0	0	1	1	1	1	X
	0	0	1	1	0	0	1	1	D
	0	1	0	1	0	1	0	1	E
000	0	0	0	0					
001	0	0	0	0					
010	0	0	0	0	1	1	1	1	
011	0	0	0	0	1	1	0	0	

ABC

To the right of these diagrams appears one (usually involving the smallest integral weights) X,Y combination of weights selected from the group. There are 8 (number to the left of the diagrams) such groups, all of which can be obtained from one another by transformations which

7. Note again the use of 1 and 0 in the truth diagrams rather than +1 and -1.

leave f invariant or by reversal of sign of all X-weights. The expression

```
A B C D E      (240)
4 4(0 0)0
```

means the following. In the full truth diagram for f, half the difference between the number of 1 entries for A = 1 and the number of 1 entries for A = 0 (or what is the same thing, the number of 1 entries for A = 1 minus 8) equals 4. Similarly these quantities for B,C,D,E are 4,0,0,0, respectively. The brackets [0] indicate that f is invariant when E is replaced by $\overline{E}$ (i.e., independent of E). The parentheses (0 0) signify that f is invariant of an interchange between C and D. The expressions ($C \leftrightarrow \overline{C}$, $D \leftrightarrow \overline{D}$) and ($A \leftrightarrow B$, $C \leftrightarrow \overline{C}$) indicate two other transformations with respect to which f is invariant. All other transformations leaving f unchanged can be obtained by combining those just given. The (240) means that there are 240 members of this symmetry class, and hence $2^5 \cdot 5!/240 = 16$ transformations leaving f invariant (including the identity transformation). Finally the X,Y weight combination to the top right of the page is usually one of the simplest (smallest sum of weights) which generate the desired function.

The page entitled "Function Index" is an aid for identifying arbitrary functions with one of those listed. For example, the knowledge that g given earlier has the following counts defined in the above paragraph,

A_0	A_1	A_2	A_3	A_4
-4	0	-4	0	0

means that it can only be type 15 or 16. Showing that brackets and parentheses can be added (that is A_1 and A_3 are interchangeable, and g is independent of A_4) means that it can only be of type 15 and nearly tells what the required correspondence of variables is. In this index parentheses and brackets are left out when the counts themselves suffice to identify the function. (Note: when the count of a variable is negative the complement of that variable is to correspond with one of the letters A,B,C,D,E with that same count. However, when the count is zero, either the variable or its complement may correspond to one of these five variables with a zero count - both possibilities must be considered in trying to reach an identification.)

Ford Motor Company
AERONUTRONIC DIVISION

TABLE I

1.	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	A B C D E (10) 8 [0] [0] [0] [0]	A B C D E 1 0 0 0 0
2.	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1	A B C D E (80) (4 4 4) [0] [0]	A B C D E 1 1 1 0 0
3.	0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1	A B C D E (32) (3 3 3 3 3)	A B C D E 1 1 1 1 1
4.	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	A B C D E (320) 6 (2 2 2) [0]	A B C D E 2 1 1 1 0
5.	0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1	A B C D E (320) (4 4) (2 2 2)	A B C D E 2 2 1 1 1
6.	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	A B C D E (160) 7 (1 1 1 1)	A B C D E 3 1 1 1 1

7.

0	0	0	0
0	0	0	0
0	0	0	0
0	1	1	1

A B C D E (960)
5 (3 3) (1 1)

A	B	C	D	E
3	2	2	1	1

8. (768)

0	0	0	0
0	0	0	0
0	0	1	0
0	1	1	1

A B C D E (960)
4 4 2 2 0
(A ↔ B, C ↔ $\overline{D}$)
(A ↔ B, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		2	1	1	0	1
Y	2	0	1	0	1	-1

a)

(8)

		0 0
0 1		

0	0	0	0	
0	0	0	0	
0	0	1	0	1 1
0	0			1 1 1 1

	2	1	1	0	1
2	0	1	0	1	-1

(96)

9. (744)

0	0	0	0
0	0	0	0
0	0	0	0
0	1	1	0

A B C D E (480)
6 (2 2) (0 0)
(D ↔ $\overline{D}$, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		1	1	1	1	1
Y	2	1	0	0	-1	-1

a)

(4)

0	1	1	1

0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	1 1 1 0

	1	1	1	1	1
2	1	0	0	-1	-1

(93)

b)

(4)

0	0	1	0

0	0	0	0	
0	0	0	0	
0	0	0	0	
0	1	0	0	1 1 1 1

	1	0	0	1	-1
2	1	1	1	-1	1

(93)

10. (384)

0	0	0	0	
0	0	1	0	
0	0	0	1	
0	1	1	1	

A B C D E (480)
(3 3) (3 3) 1
(A ↔ C, B ↔ D)

	X	A	B	C	D	E
X		1	1	0	0	1
Y	2	0	0	1	1	-1

a) (8)

				0	0	0	0	
		1	0	0	0	0		
		0	0	0	0	1		
0	0			0	1		1	1 1 1 1

	1	-1	0	0 -1
2	0	2	1	1 1

(31)

b) (4)

				0	0	0	0	
		0	0	0	0	1	0	
		0	1	0	0	0	0	
0	1			0	0		1	1 1 1 1

	1	1	0	0 1
2	0	0	1	1 -1

(34)

11. (272)

0	0	0	0
0	0	0	0
0	1	0	0
0	0	1	1

A B C D E (1920)
5 3 (1 1) 1

	X	A	B	C	D	E
X		2	0	-1	-1	1
Y	2	0	1	1	1	0

a) (4)

				0	0	0	0	
				0	0	0	0	
	1	0	0	0	0	0	1	1 1
1	1	1	1	0	0	0	0	1 1

	0	1	2	-1	1
2	2	1	-1	1	0

(32)

b) (2)

				0	0	0	0	
				0	0	0	0	
	1	0	0	0	0	0	1	1 1
0	0	0	0	0	0	1	1	1 1 1 1

	2	0	-1	-1	1
2	0	1	1	1	0

(40)

c) (2)

				0	0	0	0	
				0	0	0	0	
0	0	0	0	0	1	0	0	1 1 1 1
0	0	1	1	0	0	0	0	1 1 1 1

	2	1	1	1	0
3	0	1	-1	-1	1

(20)

d) (2)

				0	0	0	0	
				0	0	0	0	
1	1	1	0	0	0	0	0	1 0 1 1
1	0	1	0	0	0	0	1	0 1 1 1

	0	1	-1	-1	2
2	2	1	1	1	2

(12)

12. (248)

0	0	0	0	
0	0	0	0	
0	0	0	1	
0	1	1	0	

A B C D E (640)
5 3 (1 1 1)

	X	A	B	C	D	E
X		0	2	-1	-1	-1
Y	2	2	0	1	1	1

a)

(6)		0	0	0
	1	1	1	0

0	0	0	0				
0	0	0	0				
0	0	0	1		1	1	1
0	0	0	0	0	1	1	1

	1	0	2	-1	-1	
2	1	1	-1	1	1	(29)

b)

(2)	1	1	1	1
	1	1	1	0

0	0	0	0				
0	0	0	0				
0	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1

	0	2	-1	-1	-1	
2	2	0	1	1	1	(20)

c)

(2)	0	0	0	1
	0	1	1	1

0	0	0	0				
0	0	0	0				
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0

	1	1	1	1	1	
3	1	0	-1	-1	-1	(17)

13. (224)

0	0	0	0	
0	0	0	0	
1	0	0	0	
0	0	0	1	

A B C D E (320)
6 2 (0 0 0)
(C ← $\overline{C}$, D ← $\overline{D}$, E ← $\overline{E}$)

	X	A	B	C	D	E
X		2	0	1	1	1
Y	3	0	1	-1	-1	-1

a)

(12)	1	0	1	0
	1	0	1	1

0	0	0	0				
0	0	0	0				
0	0	0	0	1	1	0	1
0	0	0	0	0	1	0	1

	0	1	1	1	-2	
3	2	0	-1	-1	2	(12)

b)

(4)	0	0	0	0
	0	0	0	1

0	0	0	0				
0	0	0	0				
1	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	2	0	1	1	1	
3	0	1	-1	-1	-1	(20)

14. (224)

0	0	0	0	
0	0	0	0	
1	0	0	0	
0	1	1	1	

A B C D E (960)
4 4 2 (0 0)
(A ← B, D ← $\overline{D}$, E ← $\overline{E}$)

	X	A	B	C	D	E
X		2	1	2	1	1
Y	3	0	1	-1	-1	-1

a) (4)

1	0	0	0	0	0	0	0		
0	0	0	0	0	0	0	0	1	1	1	1
				0	1	1	1	1	1	1	1

	2	0	-1	-1	-1
3	0	2	2	1	1

(20)

b) (8)

1	0	1	0	0	0	0	0		
0	0	1	0	0	0	0	0	1	1	0	1
				0	1	0	1	1	1	1	1

	1	0	-1	1	-2
3	1	2	2	-1	2

(12)

c) (4)

0	0	0	0	0	0	0	0		
0	1	1	1	1	0	0	0	1	1	1	1
				0	0	0	0	1	1	1	1

	2	1	2	1	1
3	0	1	-1	-1	-1

(12)

15. (160)

0	0	0	0
0	0	0	0
0	0	1	1
1	1	0	0

A B C D E (240)
4 4 (0 0) 0
(C ← $\overline{C}$, D ← $\overline{D}$)
(A ← B, C ← $\overline{C}$)

	X	A	B	C	D	E
X		0	1	1	1	0
Y	2	1	0	-1	-1	0

a) (8)

0	0	1	1	0	0	0	0		
1	1	1	1	0	0	0	0	1	1	1	1
				0	0	0	0	1	1	0	0

	0	1	1	1	0
2	1	0	-1	-1	0

(20)

16. (136)

0	0	0	0
0	0	0	0
0	1	1	0
1	0	0	1

A B C D E (80)
(4 4) (0 0 0)
(C ← $\overline{C}$, D ← $\overline{D}$)
(C ← $\overline{C}$, E ← $\overline{E}$)

	X	A	B	C	D	E
X		0	0	-1	-1	-1
Y	2	1	1	1	1	1

a) (8)

1	1	1	0	0	0	0	0		
1	0	0	0	0	0	0	0	1	1	1	1
				0	0	0	1	1	1	1	1

	0	0	-1	-1	-1
2	1	1	1	1	1

(17)

17. (120)

0	0	0	1	
0	0	0	0	
0	0	0	0	
0	1	1	1	

A B C D E (960)
4 (2 2) (2 2)

	X	A	B	C	D	E
X		1	1	1	0	0
Y	3	0	-1	-1	1	1

a) (4)

			1
	0	0	0
1	1	1	
1	1	1	

0	0	0	0				1
0	0	0	0		1	1	1
0	0	0	0		0	0	0
0	0	0	0		1	1	1

	0	2	-1	1	1	
2	1	-1	1	0	0	(14)

b) (4)

	1	1	
0	1	0	
0	1	0	
0	1	0	

0	0	0	0	0		0	1
0	0	0	0		1	0	1
0	0	0	0		1	0	1
0	1	0	1		1	1	1

	0	-1	-1	1	-2	
2	1	1	1	0	2	(8)

c) (2)

			1
	0	0	0
	0	0	0
0	0	0	0

0	0	0	0				1
0	0	0	0		1	1	1
0	0	0	0		1	1	1
0	1	1	1	1	1	1	1

	1	-1	-1	0	0	
3	0	2	2	1	1	(10)

d) (2)

0	0	0	0
	0	0	0
	0	0	0
	1	1	1

0	0	0	1		1	1	1
0	0	0	0		1	1	1
0	0	0	0		1	1	1
0	0	0	0		1	1	1

	1	1	1	0	0	
3	0	-1	-1	1	1	(5)

e) (2)

1	1	1	1
1	1	1	0
1	1	1	0
1	1	1	0

0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	1	0	1	1	1

	-3	-1	-1	-2	-2	
2	3	1	1	2	2	(1)

18. (86)

0	0	0	0
0	0	0	1
0	0	1	0
1	0	0	1

A B C D E (1920)
4 2 (2 2) 0

	X	A	B	C	D	E
X		2	1	-1	-1	-2
Y	2	0	0	1	1	1

a) (2)

			0
	0	0	0
1		1	0
1	0	0	0

0	0	0	0				1
0	0	0	1		1	1	1
0	0	0	0	0		1	1
0	0	0	1	1	1	1	1

	2	1	-1	-1	-2	
2	0	0	1	1	1	(15)

18. (continued)

b) (4)

		0	0
	1	0	1
0		0	0
1	1	0	1

0	0	0	0			1	1
0	0	0	0		0	1	1
0	0	1	0	1		1	1
0	0	0	0	1	0	1	1

	1	0	2	-1	1	(8)
4	1	1	-2	2	-1	

c) (2)

	0		0
	0	1	1
0	0	1	0
1	0	1	1

0	0	0	0			1	
0	0	0	0		1	0	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	0	1

	1	1	2	2	-1	(5)
3	1	0	-1	-1	1	

d) (2)

	1		1
	1	0	1
1	1	1	1
1	1	0	1

0	0	0	0			0	0
0	0	0	0		0	1	1
0	0	0	0	0	0	1	0
0	0	0	0	1	0	1	1

	-2	1	-1	-1	2	(5)
2	2	0	1	1	-1	

e) (2)

1	1	1	1
1	1	1	1
1		1	0
1	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0		1	1
0	0	0	1	1	1	1	1

	-2	-3	-1	-1	-2		
2	2	2	1	1	1		(2)

19. (76)

0	0	0	0
0	0	0	1
0	0	0	1
0	1	1	0

A B C D E (160)
4 (2 2 2 2)

	X	A	B	C	D	E
X		1	1	1	1	1
Y	4	1	-1	-1	-1	-1

a) (8)

			0
	0	0	0
1	1	1	1
1	1	1	0

0	0	0	0				1
0	0	0	1		1	1	1
0	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1

	0	2	-1	-1	-1		
2	1	-1	1	1	1		(9)

b) (2)

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1

	-3	-1	-1	-1	-1		
2	3	1	1	1	1		(1)

c) (2)

0	0	0	0
0	0	0	1
0	0	0	1
0	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0

	1	1	1	1	1		
4	1	-1	-1	-1	-1		(1)

20. (72)

0	0	0	0	
0	1	0	0	
0	0	1	0	
0	1	1	1	

A B C D E (1920)
3 3 3 1 1
(B ↔ C, D ↔ E)

	X	A	B	C	D	E
X		-1	2	1	-2	-1
Y	3	2	0	1	2	1

a) (4)

				0
1	1	0	0	
1	1	1	0	
1	1	1		

0	0	0	0	0				1
0	0	0	0	0	1	1	1	
0	0	0	0	0	0	1	1	
0	0	0		0	1	1	1	

	-1	2	1	-2	-1	
3	2	0	1	2	1	(6)

b) (4)

1	1	1	1	
1	1	0	0	
1	1	1	0	
0	0	0	0	

0	0	0	0	0	0	0	0	
0	0	0	0	0	1	1	1	
0	0	0	0	0	0	1	1	
0	1	1	1	1	1	1	1	

	-1	-2	-3	-2	-1	
3	2	3	4	2	1	(1)

c) (4)

	1		1	
0	1	0	0	
1	1	1	1	
0	1	0		

0	0	0	0	0	0		0	
0	0	0	0	1	1	1	1	
0	0	0	0	0	0	1	0	
0	0	1		1	1	1	1	

	-1	1	-2	-1	2	
3	2	1	3	1	-1	(5)

d) (4)

0			1	
0	1	0	1	
0	1	1	1	
0	1	1	1	

0	0	0	0		0		0	
0	0	0	0	1	1	1	0	
0	0	0	0	1	0	1	0	
0	0	0	0	1	1	1	1	

	-1	1	0	1	2	
4	3	1	2	-1	-2	(3)

e) (4)

0	0		0	
0	0	0	0	
0	0	1	0	
0	0	1		

0	0	0	0			1		1
0	1	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	1	0		1	1	1	1	

	3	2	1	2	-1	
3	-1	0	1	-1	1	(3)

21. (68)

1	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	1	

A B C D E (80)
6 (0 0 0 0)
(B ↔ $\overline{B}$, C ↔ $\overline{C}$, D ↔ $\overline{D}$, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		3	1	1	1	1
Y	4	-1	-1	-1	-1	-1

a) (16)

1	0	0	0	
0	0	0	0	
1	1	1	1	
1	1	1	1	

0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	1	

	-1	3	-1	-1	-1	
4	3	-3	1	1	1	(1)

21. (continued)

b) (48)

1	0	0	0
1	0	0	0
1	1	1	0
1	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1

	-1	3	1	-2	-2
5	4	-3	-1	2	2

(1)

c) (4)

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	1

1	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	3	1	1	1	1
4	-1	-1	-1	-1	-1

(1)

22. (54)

A B C D E (960)
3 (3 3) (1 1)

0	0	0	0
0	0	0	1
0	0	0	1
1	1	1	0

	X	A	B	C	D	E
X		1	-1	-1	1	1
Y	3	0	2	2	-1	-1

a) (2)

0	0	0	0
0	0	0	1
0	0	0	1
1	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0

	1	2	2	1	1
4	1	-1	-1	-1	-1

(1)

b) (2)

			1
0	0	0	1
0	0	0	1
0	0	0	0

0	0	0	0				0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
1	1	1	0	1	1	1	1

	1	-1	-1	1	1
3	0	2	2	-1	-1

(5)

c) (4)

		1	1
0	0	1	1
0	0	1	1
0	0	1	0

0	0	0	0			0	0
0	0	0	0	1	1	0	1
0	0	0	0	1	1	0	1
1	1	0	0	1	1	1	1

	0	-1	-1	2	-1
3	1	2	2	-2	1

(3)

d) (4)

1		1	1
1	0	1	1
1	0	1	1
1	0	1	0

0	0	0	0	0		0	0
0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	1
0	1	0	0	1	1	1	1

	-2	-1	-1	1	-2
3	3	2	2	-1	2

(2)

e) (2)

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	1	1	1	1

	-3	-1	-1	-1	-1
3	4	2	2	1	1

(1)

f) (4)

			0
0	0	0	0
1	1	1	1
1	1	1	0

0	0	0	0				1
0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	1
0	0	0	0	1	1	1	1

	0	2	-1	-1	-1
3	2	-1	2	1	1

(5)

23. (52)

0	0	0	1	
0	0	0	0	
0	0	0	0	
1	1	1	1	

A B C D E (320)
(3 3 3) (1 1)

	X	A	B	C	D	E
X		1	1	1	0	0
Y	4	-1	-1	-1	1	1

a) (6)

				1
0	0	0	0	
0	0	0	0	
0	0	0	0	

0	0	0	0					1
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	

	3	-1	-1	1	1	(7)
2	-1	1	1	0	0	

b) (4)

1		1	1	
1	0	1	0	
1	0	1	0	
1	0	1	0	

0	0	0	0	0		0	1	
0	0	0	0	0	1	0	1	
0	0	0	0	0	1	0	1	
0	1	0	1	1	1	1	1	

	-1	-1	-1	1	-3	(2)
2	1	1	1	0	2	

c) (2)

0	0	0	0	
0	0	0	0	
0	0	0	0	
1	1	1	1	

0	0	0	1	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	

	1	1	1	0	0	(1)
4	-1	-1	-1	1	1	

24. (50)

0	0	0	0	
0	0	0	1	
0	1	1	0	
1	0	0	1	

A B C D E (320)
(3 3) (1 1 1)

	X	A	B	C	D	E
X		1	1	-1	-1	-1
Y	2	0	0	1	1	1

a) (2)

				0
	0	0	0	
1	1	1	0	
1	0	0	0	

0	0	0	0					1
0	0	0	1		1	1	1	
0	0	0	0	0	1	1	1	
0	0	0	1	1	1	1	1	

	1	1	-1	-1	-1	(9)
2	0	0	1	1	1	

b) (6)

	0		0	
	0	1	1	
0	0	1	0	
1	0	1	1	

0	0	0	0			1		1
0	0	0	0			1	0	1
0	1	0	0	1	1	1	1	
0	0	0	0	1	1	0	1	

	0	0	1	1	-1	(5)
3	1	1	-1	-1	2	

c) (2)

1	1	1	1	
1	1	1	1	
1	1	1	0	
1	0	0	0	

0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	1	
0	0	0	0	0	1	1	1	
0	0	0	1	1	1	1	1	

	-2	-2	-1	-1	-1	(1)
2	2	2	1	1	1	

25. (50)

0	0	0	1	
0	0	0	0	
0	0	0	0	
0	1	1	0	

A B C D E (960)
5 (1 1) (1 1)

	X	A	B	C	D	E
X		3	2	2	-1	-1
Y	3	0	-1	-1	1	1

a) (4)

0	0	0	1	
0	0	0	0	
	1	1	1	
0	1	1	1	

0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0		0	0	0	
0	0	0	0	1	1	1	0	

	1	3	-1	2	2	
4	2	-2	1	-1	-1	(2)

b) (2)

0	0	0	1	
0	0	0	1	
0	0	0	1	
0	1	1	1	

0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	0	

	1	1	1	2	2	
5	2	-1	-1	-2	-2	(1)

c) (2)

	0	0	0
	0	0	0
	0	0	0
1	1	1	0

0	0	0	1		1	1	1	
0	0	0	0		1	1	1	
0	0	0	0		1	1	1	
0	0	0	0	0	1	1	1	

	3	2	2	-1	-1	
3	0	-1	-1	1	1	(5)

d) (4)

1	1	1	1	
	0	0	0	
1	1	1	1	
1	1	1	0	

0	0	0	0	0	0	0	1	
0	0	0	0		1	1	1	
0	0	0	0	0	0	0	0	
0	0	0	0	0	1	1	1	

	-2	2	-3	-1	-1	
3	3	-1	2	1	1	(2)

e) (2)

1	1	1	1	
1	1	1	0	
1	1	1	0	
1	1	1	0	

0	0	0	0	0	0	0	1	
0	0	0	0	0	0	0	1	
0	0	0	0	0	0	0	1	
0	0	0	0	0	1	1	1	

	-3	-1	-1	-2	-2	
3	4	1	1	2	2	(1)

f) (4)

	0	1	1	
	0	1	0	
	0	1	0	
0	0	1	0	

0	0	0	0		1	0	1	
0	0	0	0		1	0	1	
0	0	0	0		1	0	1	
0	1	0	0	1	1	1	1	

	1	-1	-1	2	-2	
3	1	1	1	-1	2	(5)

26. (44)

0	0	0	0
0	0	0	1
1	0	0	1
1	0	0	1

A B C D E (1920)
3 3 1 (1 1)

	X	A	B	C	D	E
X		3	1	1	2	2
Y	3	-1	1	0	-1	-1

a) (4)

	0	1	
	0	1	1
1	0	1	1
1	0	1	1

0	0	0	0		1	0		
0	0	0	0		1	0	1	
0	0	0	0	1	1	0	1	
0	0	0	0	1	1	0	1	

	-1	2	1	3	-2	
2	1	0	0	-1	1	(7)

26. (continued)

b) (2)

	0	0	0
	0	0	0
1	0	0	0
1	0	0	0

0	0	0	0		1	1	1
0	0	0	1		1	1	1
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1

	2	1	0	-1	-1
4	-1	1	1	2	2

(4)

c) (2)

1	1	1	
1	1	1	1
1	0	0	0
1	0	0	0

0	0	0	0	0	0	0	
0	0	0	0	0	0	0	1
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1

	-1	-3	1	-2	-2
2	1	2	0	1	1

(2)

d) (2)

0	0	0	
0	0	0	1
0	0	0	1
0	0	0	1

0	0	0	0	1	1	1	
0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1

	3	1	1	2	2
3	-1	1	0	-1	-1

(2)

27. (40)

0	0	0	0
1	0	0	0
0	0	0	1
1	0	0	1

A B C D E (960)
4 2 2 (0 0)
(B ↔ C, D ↔ $\bar{D}$, E ↔ $\bar{E}$)

	X	A	B	C	D	E
X		2	1	0	1	1
Y	3	-1	0	1	-1	-1

a) (8)

	0	1	
1	0	1	0
1	0	1	1
1	0	1	1

0	0	0	0		1	0	
0	0	0	0	1	1	0	1
0	0	0	0	0	1	0	1
0	0	0	0	1	1	0	1

	-1	1	0	1	-2
3	2	0	1	-1	2

(3)

b) (4)

	1	1	1
1	1	1	1
0	0	0	1
0	0	0	1

0	0	0	0		0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1

	-1	-2	0	1	1
3	2	3	1	-1	-1

(2)

c) (4)

0	0	0	
0	0	0	0
0	0	0	1
0	0	0	1

0	0	0	0	1	1	1	
1	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1

	2	1	0	1	1
3	-1	0	1	-1	-1

(2)

28. (40)

0	0	1	0
0	0	0	0
0	0	0	0
0	1	1	1

A B C D E (1920)
4 (2 2) 2 0

	X	A	B	C	D	E
X		2	-1	-1	0	-1
Y	4	-1	2	2	1	1

a)

		1	1	1
	0	1	0	1
(2)	0	1	0	1
	0	1	0	1

0	0	0	0			0	1	0
0	0	0	0		1	0	1	0
0	0	0	0		1	0	1	0
0	0	1	0		1	1	1	1

	-1	-1	-1	0	2	
4	3	2	2	1	-3	(2)

b)

		1	1	
0	0	0	0	
(4)	1	1	1	1
0	1	1	1	

0	0	0	0				1	0
0	0	0	0		1	1	1	1
0	0	0	0		0	0	0	0
0	0	0	0		1	1	1	1

	-1	2	-2	1	1	
3	2	-1	2	0	-1	(3)

c)

		1	0	
0	0	0	0	
(4)	1	1	1	1
1	1	1	1	

0	0	0	0				1	1
0	0	0	0		1	1	1	1
0	0	0	0		0	0	0	0
0	0	0	0		0	1	1	1

	-1	2	-1	0	-1	
4	3	-2	2	1	1	(3)

d)

1	1	1	1	
	1	1	0	0
(2)	1	1	0	0
	0	1	0	0

0	0	0	0		0	0	1	0
0	0	0	0		0	0	1	1
0	0	0	0		0	0	1	1
0	0	1	1		1	1	1	1

	-1	-2	-2	-3	1	
3	2	2	2	3	-1	(1)

e)

1	1	1	0	
	1	1	0	0
(2)	1	1	0	0
	1	1	0	0

0	0	0	0		0	0	1	1
0	0	0	0		0	0	1	1
0	0	0	0		0	0	1	1
0	0	1	1		0	1	1	1

	-1	-1	-1	-3	-1	
4	3	2	2	5	1	(1)

f)

		1	0	
0	0	0	0	
(2)	0	0	0	0
0	0	0	0	

0	0	0	0				1	1
0	0	0	0		1	1	1	1
0	0	0	0		1	1	1	1
0	1	1	1		1	1	1	1

	2	-1	-1	0	-1	
4	-1	2	2	1	1	(3)

g)

0	0	0	0	
	0	0	0	0
(2)	0	0	0	0
	0	1	1	1

0	0	1	0		1	1	1	1
0	0	0	0		1	1	1	1
0	0	0	0		1	1	1	1
0	0	0	0		1	1	1	1

	3	2	2	1	1	
4	-1	-1	-1	1	-1	(1)

29. (40)

	0	0	0	1		
	0	0	0	0		
	0	0	0	0		
	1	0	0	1		

	A	B	C	D	E	(480)
	5	(1	1)	(1	1)	
	(B ← D, C ← E)					

	X	A	B	C	D	E
X		-1	3	-2	2	-1
Y	4	3	-2	2	-1	1

a) (8)

	1	1	1	1	0	0	0	1		
	1	0	1	0	0	1	0	1		
	1	0	1	0	0	1	0	1		
	1	0	0	0	0	0	0	1		

	-1	-2	-2	-1	-3	
4	3	2	2	1	3	(1)

b) (16)

		0	1	0			1	0	1	
	0	0	1	0	1	1	1	1		
	1	1	1	1	0	1	0	0		
	1	0	1	1	0	1	0	1		

	-1	3	-2	2	-1	
4	3	-2	2	-1	1	(2)

30. (32)

	0	1	0	0		
	0	0	0	0		
	0	0	0	0		
	0	0	1	1		

	A	B	C	D	E	(640)
	5	(1	1	1)	1	

	X	A	B	C	D	E
X		3	-1	-1	-1	1
Y	3	-1	1	1	1	0

a) (2)

	1	1	1	0	0	0	0	0		
	1	0	1	0	0	0	0	0		
	1	0	1	0	0	0	0	0		
	1	0	1	0	0	0	0	1		

	-1	-1	-1	-1	-3	
3	2	1	1	1	3	(1)

b) (6)

		1	0	0			1	1	1	
	1	1	0	0	0	0	1	1		
	1	1	0	0	0	0	1	1		
	1	1	1	1	0	0	1	1		

	-1	2	2	-3	1	
3	2	-1	-1	2	0	(2)

c) (6)

		1	0	0			1	1	1	
	0	0	0	0	1	1	1	1		
	1	1	1	1	0	0	0	0		
	1	1	1	1	0	0	1	1		

	-1	3	-1	-1	1	
3	2	-2	1	1	0	(2)

d) (2)

		1	0	0			1	1	1	
	0	0	0	0	1	1	1	1		
	0	0	0	0	1	1	1	1		
	0	0	0	0	1	1	1	1		

	3	-1	-1	-1	1	
3	-1	1	1	1	0	(2)

30. (continued)

e)

	0	0	0	0	0	1	0	0	1 1 1 1
	0	0	0	0	0	0	0	0	1 1 1 1
(2)	0	0	0	0	0	0	0	0	1 1 1 1
	0	0	1	1	0	0	0	0	1 1 1 1

	2	1	1	1	0
4	-1	-1	-1	-1	1

(1)

31. (32)

0	0	0	0
1	0	0	0
0	0	0	1
0	1	1	0

A B C D E (960)
4 2 2 (0 0)
(B ↔ C,D ↔ $\overline{D}$,E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		-1	-1	3	2	2
Y	3	2	1	-1	-1	-1

a)

0			1	
1	1	1	1	
0	0	0	1	
0	1	1	1	

0	0	0	0	1			0		
0	0	0	0	1	0	0	0		
0	0	0	0	1	1	1	1		
0	0	0	0	1	1	1	0		

	-1	-1	3	2	2
3	2	1	-1	-1	-1

(4)

(4)

b)

1		1	1	
1	0	1	1	
0	0	1	1	
0	0	1	0	

0	0	0	0	0		0	0		
0	0	0	0	1	1	0	0		
0	0	0	0	1	1	0	1		
0	1	0	0	1	1	1	1		

	-1	-2	-1	3	-2
3	2	2	1	-2	1

(2)

(8)

32. (32)

0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	1

A B C D E (1920)
4 2 2 0 0
(B ↔ C, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		1	2	-1	2	1
Y	4	1	-1	2	-2	-1

a)

1	1			
1	1	0	0	
0	1	0	0	
0	0	0	0	

0	0	0	0	0	0				
0	0	0	0	1	0	1	1		
0	0	0	0	1	1	1	1		
0	0	1	1	1	1	1	1		

	1	-2	-1	-2	1
4	1	3	2	2	-1

(3)

(4)

b)

1	1	1	1	
1	1	1	1	
0	1	1	1	
0	0	1	1	

0	0	0	0	0	0	0	0		
0	0	0	0	1	0	0	0		
0	0	0	0	1	1	0	0		
0	0	0	0	1	1	1	1		

	-3	-2	-1	2	1
4	5	3	2	-2	-1

(1)

(4)

32. (continued)

c)
(4)

0	0	0	0		0	0	0	0	1	1	1	1		1	2	3	-2	1
1	1	0	0		0	0	0	0	1	0	1	1	5	2	-1	-2	2	-1
0	1	0	0		0	0	0	0	1	1	1	1						
1	1	1	1		0	0	0	0	0	0	1	1						

(1)

d)
(4)

0	0				0	0	0	0	1	1				1	2	-1	2	1
0	0	0	0		1	0	0	0	1	1	1	1	4	1	-1	2	-2	-1
0	1	1	1		0	0	0	0	1	1	0	0						
0	0	1	1		0	0	0	0	1	1	1	1						

(3)

33. (28)

A B C D E (320)
2 (2 2 2) [0]

0	0	0	0	
0	0	1	1	
0	0	1	1	
1	1	0	0	

	X	A	B	C	D	E
X		0	1	1	1	0
Y	3	1	-1	-1	-1	0

a)
(6)

		0	0		0	0	0	0			1	1		1	2	-1	-1	0
0	0	0	0		0	0	1	1	1	1	1	1	2	0	-1	1	1	0
1	1	1	1		0	0	0	0	0	0	1	1						
1	1	0	0		0	0	0	0	1	1	1	1						

(4)

b)
(2)

1	1	1	1		0	0	0	0	0	0	0	0		-2	-1	-1	-1	0
1	1	1	1		0	0	0	0	0	0	1	1	2	2	1	1	1	0
1	1	1	1		0	0	0	0	0	0	1	1						
1	1	0	0		0	0	0	0	1	1	1	1						

(1)

c)
(2)

0	0	0	0		0	0	0	0	1	1	1	1		0	1	1	1	0
0	0	1	1		0	0	0	0	1	1	1	1	3	1	-1	-1	-1	0
0	0	1	1		0	0	0	0	1	1	1	1						
1	1	1	1		0	0	0	0	1	1	0	0						

(1)

34. (24)

A B C D E (480)
2 (2 2) (0 0)
(D ← $\overline{D}$, E ← $\overline{E}$)

0	0	0	0	
1	0	0	1	
1	0	0	1	
1	0	0	1	

	X	A	B	C	D	E
X		1	0	0	1	1
Y	4	-1	1	1	-2	-2

a)
(4)

	0	1			0	0	0	0		1	0			-1	1	1	2	-2
1	0	1	1		0	0	0	0	1	1	0	1	2	1	0	0	-1	1
1	0	1	1		0	0	0	0	1	1	0	1						
1	0	1	1		0	0	0	0	1	1	0	1						

(4)

34. (continued)

b)

0	0	0		
0	0	0	1	
0	0	0	1	
0	0	0	1	

(4)

0	0	0	0	1	1	1			
1	0	0	0	1	1	1	1		
1	0	0	0	1	1	1	1		
1	0	0	0	1	1	1	1		

	1	0	0	1	1	
4	-1	1	1	-2	-2	(2)

35. (24)

0	0	0	1	
0	0	0	0	
0	0	0	1	
1	1	1	0	

A B C D E (1920)
3 3 1 (1 1)

	X	A	B	C	D	E
X		3	2	2	-1	-1
Y	3	-1	0	-1	1	1

a)

0	0	0	1	
	1	1	1	
0	0	0	1	
1	1	1	1	

(2)

0	0	0	0	1	1	1	1		
0	0	0	0		0	0	0		
0	0	0	0	1	1	1	1		
0	0	0	0	1	1	1	0		

	-1	1	3	2	2	
4	2	1	-2	-1	-1	(2)

b)

1	1	1	1	
1	1	1	1	
0	0	0	0	
1	1	1	0	

(2)

0	0	0	0	0	0	0	1		
0	0	0	0	0	0	0	0		
0	0	0	1	1	1	1	1		
0	0	0	0	1	1	1	1		

	-2	-3	2	-1	-1	
3	2	3	-1	1	1	(1)

c)

1	1	1	1	
	0	0	0	
1	1	1	1	
1	1	1	0	

(2)

0	0	0	0	0	0	0	1		
0	0	0	0		1	1	1		
0	0	0	0	0	0	0	1		
0	0	0	0	1	1	1	1		

	-2	2	-3	-1	-1	
3	2	0	2	1	1	(2)

d)

1	0	1	1	
	0	1	0	
1	0	1	1	
1	0	1	0	

(4)

0	0	0	0	0	1	0	1		
0	0	0	0		1	0	1		
0	0	0	0	0	1	0	1		
0	1	0	0	1	1	1	1		

	-1	1	-2	2	-3	
4	2	1	2	-1	3	(2)

e)

0	0	0	1	
0	0	0	0	
0	0	0	1	
0	0	0	0	

(2)

0	0	0	0	1	1	1	1		
0	0	0	0	1	1	1	1		
0	0	0	0	1	1	1	1		
1	1	1	0	1	1	1	1		

	2	0	-1	1	1	
5	-2	2	2	-1	-1	(1)

f)

0	0	0	0	
	0	0	0	
0	0	0	0	
1	1	1	0	

(2)

0	0	0	1	1	1	1	1		
0	0	0	0		1	1	1		
0	0	0	1	1	1	1	1		
0	0	0	0	1	1	1	1		

	3	2	2	-1	-1	
3	-1	0	-1	1	1	(2)

36. (24)

0	0	0	0	
0	0	0	1	
1	0	0	1	
0	1	1	0	

A B C D E (320)
(3 3) (1 1 1)

	X	A	B	C	D	E
X		-1	-1	3	-2	-2
Y	2	1	1	-1	1	1

a)

1			0	
1	1	1	1	
1	0	0	0	
1	1	1	0	

(6)

0	0	0	0	0				1
0	0	0	0	0	0	0	1	
0	0	0	1	1	1	1	1	
0	0	0	0	0	1	1	1	

	-1	-1	3	-2	-2	
2	1	1	-1	1	1	(4)

37. (20)

1	0	0	0	
0	0	0	0	
0	0	0	1	
0	1	1	0	

A B C D E (640)
4 2 (0 0 0)

	X	A	B	C	D	E
X		2	2	1	1	1
Y	3	0	-1	-1	-1	-1

a)

1	0	0	0	
0	0	0	0	
1	1	1	1	
1	1	1	0	

(2)

0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	0	0	0	1	
0	0	0	0	0	1	1	1	

	0	2	-1	-1	-1	
4	2	-2	1	1	1	(1)

b)

1	0	0	0	
1	1	1		
0	0	0	0	
1	1	1	0	

(6)

0	0	0	0	1	1	1	1	
0	0	0	0	0	0	0		
0	0	0	1	1	1	1	1	
0	0	0	0	0	1	1	1	

	1	-1	3	-2	-2	
3	1	1	-2	1	1	(2)

c)

0	0	0	0	
0	0	0		
0	0	0	1	
0	1	1	1	

(2)

1	0	0	0	1	1	1	1	
0	0	0	0	1	1	1		
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	0	

	2	2	1	1	1	
3	0	-1	-1	-1	-1	(2)

d)

1	1	1	1	
1	1	1	1	
0	0	0	1	
0	1	1	1	

(2)

0	0	0	0	1	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	0	

	-2	-2	1	1	1	
3	3	2	-1	-1	-1	(1)

38. (20)

0	0	0	1
0	0	0	0
0	1	0	0
1	0	1	1

A B C D E (1920)
3 3 1 1 1
(A ↔ B, D ↔ E)

	X	A	B	C	D	E
X		3	-1	-2	-1	2
Y	4	-1	2	2	1	-1

a) (4)

	0	1	1
1	1	1	1
0	0	0	0
1	0	1	1

0	0	0	0		1	0	1
0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1
0	0	0	0	1	1	1	1

-1	-2	3	2	-1	(2)
4	2	3	-2	-1	1

b) (4)

	1	0	1
0	0	0	0
0	1	0	0
0	0	0	0

0	0	0	0		0	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
1	0	1	1	1	1	1	1

3	-1	-2	-1	2	(2)
4	-1	2	2	1	-1

c) (4)

1	1	1	1
1	1	0	0
1	1	0	0
1	0	0	0

0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	1
0	0	1	1	1	1	1	1

-1	-2	-2	-3	-1	(1)
4	2	3	2	3	1

39. (16)

1	1	0	0
0	0	0	0
0	0	0	0
0	0	1	1

A B C D E (160)
4 (0 0 0) [0]
(B ↔ $\overline{B}$, C ↔ $\overline{C}$, D ↔ $\overline{D}$)

	X	A	B	C	D	E
X		2	1	1	1	0
Y	3	-1	-1	-1	-1	0

a) (12)

1	1	0	0
1	1	0	0
1	1	0	0
1	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1

-1	1	1	-2	0	(1)
3	2	-1	-1	2	0

b) (4)

0	0	0	0
0	0	0	0
0	0	0	0
0	0	1	1

1	1	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

2	1	1	1	0	(1)
3	-1	-1	-1	-1	0

40. (16)

0	0	1	0
1	0	0	0
0	0	0	1
0	1	1	1

A B C D E (960)
(2 2) (2 2) 0

	X	A	B	C	D	E
X		-1	-1	2	2	1
Y	4	2	2	-1	-1	-1

a) (2)

0		1	1
1	1	1	1
0	0	0	1
0	1	1	1

0	0	0	0	1		1	0
0	0	0	0	1	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

		-1	-1	2	2	1
4	2	2	-1	-1	-1	(2)

b) (2)

1	1	1	1
1	1	0	1
0	1	0	1
0	1	0	1

0	0	0	0	0	0	1	0
0	0	0	0	1	0	1	0
0	0	0	0	1	0	1	1
0	0	1	0	1	1	1	1

		-2	-2	-1	-1	3
3	2	2	1	1	-2	(1)

c) (2)

1	1	1	1
1	1	0	0
1	1	0	1
0	1	0	0

0	0	0	0	0	0	1	0
0	0	0	0	1	0	1	1
0	0	0	0	0	0	1	1
0	0	1	1	1	1	1	1

		-1	-1	-2	-2	1
4	2	2	3	3	-1	(1)

d) (4)

1		1	0
1	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0	0		1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	1

		3	-2	-1	-1	-2
3	-1	2	1	1	1	(2)

41. (16)

1	0	0	0
0	0	0	0
0	0	0	0
0	1	1	1

A B C D E (960)
4 (2 2) (0 0)

	X	A	B	C	D	E
X		3	2	2	1	1
Y	4	-1	-1	-1	-1	-1

a) (4)

1	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

0	0	0	0	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	1	1	0	0
0	1	0	0	1	1	1	1

		-1	-1	-1	3	-1
5	3	2	2	-4	1	(1)

b) (4)

1	1	1	1
0	0	0	0
1	1	1	1
0	1	1	1

0	0	0	0	1	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1

		-2	2	-3	1	1
4	3	-1	3	-1	-1	(1)

41. (continued)

c)
(4)

1	0	0	0
0	0	0	0
1	1	1	1
1	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1

	-1	3	-1	-1	-1
5	3	-3	2	1	1

(1)

d)
(2)

0	0	0	0
0	0	0	0
0	0	0	0
0	1	1	1

1	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	3	2	2	1	1
4	-1	-1	-1	-1	-1

(1)

e)
(2)

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1

	3	-1	-1	-1	-1
5	-2	2	2	1	1

(1)

42. (16)

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0

A B C D E (120)
4 (0 0) (0 0)
(B ↔ D, C ↔ E)
(B ↔ $\overline{B}$, C ↔ $\overline{C}$)

	X	A	B	C	D	E
X		-1	2	2	1	1
Y	4	3	-2	-2	-1	-1

a)
(16)

0	0	0	0
0	1	1	1
0	1	1	1
1	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

	-1	2	2	1	1
4	3	-2	-2	-1	-1

(1)

43. (14)

0	1	1	1
0	0	0	0
0	0	0	0
1	1	1	1

A B C D E (320)
(1 1 1) (1 1)

	X	A	B	C	D	E
X		1	1	1	0	0
Y	5	-2	-2	-2	1	1

a)
(6)

	1	1	1
0	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0		1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1

	3	-2	-2	1	1
2	-1	1	1	0	0

(2)

b)
(2)

0	0	0	0
0	0	0	0
0	0	0	0
1	1	1	1

0	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	1	1	1	0	0
5	-2	-2	-2	1	1

(1)

44. (14)

0	0	0	1
0	0	0	1
1	0	0	0
1	0	0	1

A B C D E (1920)
3 1 1 (1 1)

	X	A	B	C	D	E
X		3	-1	1	2	2
Y	3	-1	1	0	-1	-1

a) (2)

1	1	1	1
1	1	1	1
1	0	0	0
1	0	0	0

0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1

	-1	-2	0	-1	-1	
5	3	4	1	2	2	(1)

b) (4)

0	0	1	1
0	0	1	1
1	0	1	1
1	0	1	1

0	0	0	0	1	1	0	1
0	0	0	0	1	1	0	1
0	0	0	0	1	1	0	0
0	0	0	0	1	1	0	1

	-1	1	0	2	-1	
5	3	-1	1	-3	2	(1)

c) (4)

1	0	1	1
1	0	1	1
1	0	1	0
1	0	1	1

0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	1
0	0	0	0	1	1	0	1
0	0	0	0	1	1	0	1

	-2	-1	1	2	-3	
3	2	1	0	-1	2	(1)

d) (2)

0	0	0	1
0	0	0	1
0	0	0	0
0	0	0	1

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1

	3	-1	1	2	2	
3	-1	1	0	-1	-1	(1)

e) (2)

0	0	0	0
0	0	0	0
1	0	0	0
1	0	0	0

0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1

	2	1	0	-1	-1	
5	-2	-1	1	2	2	(1)

45. (14)

1	0	0	0
0	0	0	0
0	0	0	1
0	1	1	1

A B C D E (320)
(3 3) (1 1 1)

	X	A	B	C	D	E
X		2	2	1	1	1
Y	4	-1	-1	-1	-1	-1

a) (2)

1	1	1	1
1	1	1	1
0	0	0	1
0	1	1	1

0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	-2	-2	1	1	1	
4	3	3	-1	-1	-1	(1)

45. (continued)

b) (6)

1	0	0	0	0	0	0	0	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1
1	1	1	1	0	0	0	0	0	1	1	1

	-1	-1	3	-1	-1	(1)
4	2	2	-3	1	1	

c) (4)

1	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1

	3	-1	-1	-1	-1	(1)
4	-2	2	1	1	1	

d) (2)

0	0	0	0	1	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0	1	1	1	1
0	1	1	1	0	0	0	0	1	1	1	1

	2	2	1	1	1	(1)
4	-1	-1	-1	-1	-1	

46. (12)

1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1

A B C D E (960)
2 (2 2) (0 0)

	X	A	B	C	D	E
X		1	2	2	1	1
Y	3	0	-1	-1	-1	-1

a) (2)

1	1	1	1	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0	1	1	1	1

	1	-2	-2	1	1	(1)
3	0	2	2	-1	-1	

b) (4)

1	0	1	1	0	0	0	0	1	1	0	0
0	0	1	1	0	0	0	0	1	1	0	1
0	0	1	1	0	0	0	0	1	1	0	1
0	0	1	0	1	1	0	0	1	1	1	1

	0	-1	-1	2	-1	(1)
4	1	2	2	-3	1	

c) (2)

0	0	0	0	1	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0	1	1	1	1
1	1	1	1	0	0	0	0	1	1	1	0

	1	2	2	1	1	(1)
3	0	-1	-1	-1	-1	

d) (4)

1	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	1	1	1	1

	0	2	-1	-1	-1	(1)
4	1	-2	2	1	1	

47. (12)

1	0	0	0	
0	0	0	1	
0	0	0	1	
0	1	1	1	

A B C D E (32)
(2 2 2 2 2)

	X	A	B	C	D	E
X		1	1	1	1	1
Y	4	-1	-1	-1	-1	-1

a) (10)

1	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1

	3	-1	-1	-1	-1
3	-2	1	1	1	1

(1)

b) (2)

0	0	0	0	1	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0	1	1	1	1
0	1	1	1	0	0	0	0	1	1	1	1

	1	1	1	1	1
4	-1	-1	-1	-1	-1

(1)

48. (12)

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	1

A B C D E (480)
3 (1 1) (1 1)
(B ↔ D, C ↔ E)

	X	A	B	C	D	E
X		-1	2	2	-1	-1
Y	3	2	-1	-1	1	1

a) (4)

		0	0	0	0	0	0	1	1	1
1	1	1	0	0	0	0	0	1	1	1
1	1	1	0	0	0	0	0	1	1	1
1	1	1	1	0	0	0	0	0	0	1

	-1	2	2	-1	-1
3	2	-1	-1	1	1

(2)

b) (4)

1	1	1	0	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	1	1	1
1	1	1	0	0	0	0	0	1	1	1
1	0	0	0	0	0	0	1	0	1	1

	-1	-1	-1	-2	-2
3	2	1	1	2	2

(1)

49. (12)

1	1	1	0
0	0	0	0
0	0	0	0
0	1	1	1

A B C D E (480)
2 (0 0) (0 0)
(B ↔ $\overline{B}$, C ↔ $\overline{C}$, D ↔ $\overline{D}$, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		3	2	2	1	1
Y	5	-2	-2	-2	-1	-1

a) (8)

1	1	1	1	0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1	1	1

	-2	2	-3	1	1
5	3	-2	3	-1	-1

(1)

49. (continued)

b)

0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	1	1	1	1	1	1	1	

(4)

1	1	1	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	

	3	2	2	1	1
5	-2	-2	-2	-1	-1

(1)

50. (12)

1	0	0	0
0	0	0	1
1	0	0	1
1	0	0	1

A B C D E (960)
2 2 0 (0 0)
(C ↔ $\overline{C}$, D ↔ $\overline{D}$, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		3	1	1	2	2
Y	5	-2	1	-1	-2	-2

a)

1	0	1	0	0	0	0	0	
1	0	1	1	0	0	0	0	
1	0	1	1	0	0	0	0	
1	0	1	1	0	0	0	0	

(8)

0	0	0	0	1	1	0	1	
0	0	0	0	0	1	0	1	
0	0	0	0	1	1	0	1	
0	0	0	0	1	1	0	1	

	-2	1	1	2	-3
5	3	1	-1	-2	3

(1)

b)

0	0	0	0	1	0	0	0	
0	0	0	1	0	0	0	0	
0	0	0	1	0	0	0	0	
0	0	0	1	0	0	0	0	

(4)

1	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
1	0	0	0	1	1	1	1	
1	0	0	0	1	1	1	1	

	3	1	1	2	2
5	-2	1	-1	-2	-2

(1)

51. (12)

0	0	0	1
0	0	0	1
0	0	0	1
1	1	1	0

A B C D E (480)
2 (2 2) (2 2)
(B ↔ D, C ↔ E)

	X	A	B	C	D	E
X		3	2	2	-1	-1
Y	5	-2	-1	-1	2	2

a)

1	1	1	1	0	0	0	1	
0	0	0	0	1	1	1	1	
1	1	1	1	0	0	0	1	
1	1	1	0	1	1	1	1	

(8)

0	0	0	0	0	0	0	1	
0	0	0	1	1	1	1	1	
0	0	0	0	0	0	0	1	
0	0	0	0	1	1	1	1	

	-2	2	-3	-1	-1
5	3	-1	4	2	2

(1)

b)

0	0	0	0	0	0	0	1	
0	0	0	0	0	0	0	1	
0	0	0	0	0	0	0	1	
1	1	1	0	1	1	1	1	

(4)

0	0	0	1	1	1	1	1	
0	0	0	1	1	1	1	1	
0	0	0	1	1	1	1	1	
0	0	0	0	1	1	1	1	

	3	2	2	-1	-1
5	-2	-1	-1	2	2

(1)

52. (12)

0	0	0	0	
1	1	0	0	
0	0	1	1	
1	0	0	1	

A	B	C	D	E	(1920)
2	2	2	0	0	

$(B \rightarrow C,\ D \rightarrow \overline{D},\ E \rightarrow \overline{E})$

	X	A	B	C	D	E
X		3	2	-1	2	1
Y	5	-2	-1	2	-2	-1

a) (4)

1	1	1	1		0	0	0	0	
1	1	1	1		0	0	0	0	
0	0	1	1		0	0	0	0	
0	0	0	1		1	0	0	0	

					0	0	0	0	
					1	1	0	0	
					1	1	1	1	
					1	1	1	1	

		-2	-3	-1	2	1	
5	3	4	2	-2	-1		(1)

b) (4)

1	1	0	0		0	0	1	1	
1	1	0	0		1	1	1	1	
1	1	1	1		0	0	1	1	
1	1	0	1		1	0	1	1	

		-2	2	-1	-3	1	
5	3	-1	2	3	-1		(1)

c) (4)

0	0	0	0		1	1	1	1	
0	0	0	0		1	1	1	1	
0	0	1	1		1	1	1	1	
0	0	0	1		1	1	1	1	

		3	2	-1	2	1	
5	-2	-1	2	-2	-1		(1)

53. (12)

0	0	0	1	
1	0	0	0	
0	0	0	1	
1	0	1	1	

A	B	C	D	E	(1920)
2	2	2	2	0	

$(C \rightarrow D,\ E \rightarrow \overline{E})$

	X	A	B	C	D	E
X		3	1	-1	2	2
Y	5	-2	1	2	-1	-2

a) (4)

1	1	0	1		0	0	0	0	
1	1	0	0		0	0	0	0	
1	1	0	1		0	0	0	0	
1	1	0	1		0	0	1	0	

					0	0	1	1	
					1	0	1	1	
					0	0	1	1	
					1	0	1	1	

		-2	1	-1	-3	2	
5	3	1	2	4	-2		(1)

b) (4)

0	1	0	1		0	0	0	0	
1	1	0	1		0	0	0	0	
0	1	0	1		0	0	0	0	
1	1	1	1		0	0	0	0	

					1	0	1	1	
					1	0	1	0	
					1	0	1	1	
					1	0	1	1	

		-2	1	2	-1	3	
5	3	1	-1	2	-3		(1)

c) (4)

0	0	0	1		0	0	0	0	
0	0	0	0		1	0	0	0	
0	0	0	1		0	0	0	0	
0	0	0	1		0	0	0	0	

					1	1	1	1	
					1	1	1	1	
					1	1	1	1	
					1	1	1	1	

		3	1	-1	2	2	
5	-2	1	2	-1	-2		(1)

54. (12)

1	0	0	0
0	0	0	0
0	1	0	0
0	0	1	1

A	B	C	D	E (1920)
4	2	(0	0)	0

	X	A	B	C	D	E
X		3	-1	-2	-2	1
Y	5	-1	2	2	2	-1

a) (4)

1	1	0	0
1	1	1	1
0	1	0	0
1	1	1	1

0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1

	-2	-1	3	-2	1
5	4	2	-3	2	-1

(1)

b) (4)

1	0	0	0
1	1	0	0
1	1	0	0
1	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1

	-1	2	2	-3	-1
5	3	-1	-2	3	1

(1)

c) (2)

1	0	1	1
1	1	1	1
0	0	0	0
0	0	1	1

0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	-1	-3	2	2	-1
5	3	4	-2	-2	1

(1)

d) (2)

1	1	0	0
0	0	0	0
0	1	0	0
0	0	0	0

0	0	0	0	1	0	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	1	1	1	1	1	1

	3	-1	-2	-2	1
5	-1	2	2	2	-1

(1)

55. (12)

1	0	0	1
0	0	0	0
0	0	0	0
1	1	1	1

A	B	C	D	E (160)
(2	2	2)	(0	0)

($D \leftrightarrow \overline{D}$, $E \leftrightarrow \overline{E}$)

	X	A	B	C	D	E
X		3	-2	-2	1	-1
Y	5	-2	3	3	-1	1

a) (12)

1	0	1	1
0	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0	1	1	0	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1

	3	-2	-2	1	-1
5	-2	3	3	-1	1

(1)

56. (10)

1	0	0	0	
0	0	0	1	
0	0	0	1	
0	1	1	0	

A B C D E (160)
3 (1 1 1 1)

	X	A	B	C	D	E
X		1	1	1	1	1
Y	3	0	-1	-1	-1	-1

a) (8)

1	0	0	0	
0	0	0	0	
1	1	1	1	
1	1	1	0	

0	0	0	0	1	1	1	1	
0	0	0	1	1	1	1	1	
0	0	0	0	0	0	0	1	
0	0	0	0	0	1	1	1	

	0	2	-1	-1	-1	(1)
3	1	-2	1	1	1	

b) (2)

0	0	0	0	
0	0	0	1	
0	0	0	1	
0	1	1	1	

1	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	0	

	1	1	1	1	1	(1)
3	0	-1	-1	-1	-1	

57. (10)

0	1	1	0	
0	0	0	0	
0	0	0	0	
0	1	1	1	

A B C D E (960)
3 (1 1) (1 1)

	X	A	B	C	D	E
X		3	-2	-2	-1	-1
Y	4	-1	2	2	1	1

a) (4)

0	1	1	1	
0	0	0	0	
1	1	1	1	
0	1	1	1	

0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	1	
0	0	0	0	0	0	0	0	
0	0	0	0	1	1	1	1	

	-1	2	-2	1	1	(1)
5	3	-2	3	-1	-1	

b) (4)

1	1	1	0	
0	0	0	0	
1	1	1	1	
1	1	1	1	

0	0	0	0	0	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	0	0	0	0	
0	0	0	0	0	1	1	1	

	-2	3	-2	-1	-1	(1)
4	3	-2	2	1	1	

c) (2)

1	1	1	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	

0	0	0	0	0	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	1	1	1	1	1	1	1	

	3	-2	-2	-1	-1	(2)
4	-1	2	2	1	1	

58. (10)

0	0	0	1	
0	1	1	0	
0	1	1	0	
0	1	1	0	

A B C D E (960)
1 (1 1) (1 1)

	X	A	B	C	D	E
X		-1	1	1	2	2
Y	5	2	-1	-1	-2	-2

a) (2)

1	0	0	0	
1	1	1	0	
1	1	1	0	
1	1	1	0	

0	0	0	1	0	1	1	1	
0	0	0	0	0	1	1	1	
0	0	0	0	0	1	1	1	
0	0	0	0	0	1	1	1	

	-1	1	1	-2	-2	(1)
5	2	-1	-1	3	3	

b) (2)

1	1	1	1	
1	1	1	0	
1	1	1	0	
1	1	1	0	

0	0	0	0	0	0	0	1	
0	0	0	0	0	1	1	1	
0	0	0	0	0	1	1	1	
0	0	0	0	0	1	1	1	

	-3	-1	-1	-2	-2	(1)
4	3	1	1	2	2	

c) (2)

0	0	0	1	
0	1	1	1	
0	1	1	1	
0	1	1	1	

0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	0	

	-1	1	1	2	2	(1)
5	2	-1	-1	-2	-2	

d) (4)

0	0	1	1	
0	0	1	0	
0	0	1	0	
0	0	1	0	

0	0	0	0	1	1	0	1	
0	1	0	0	1	1	1	1	
0	1	0	0	1	1	1	1	
0	1	0	0	1	1	1	1	

	2	-1	-1	3	-2	(1)
4	-1	1	1	-2	2	

59. (10)

0	0	0	0	
1	0	0	1	
0	1	1	0	
0	0	0	1	

A B C D E (1920)
3 1 1 (1 1)

	X	A	B	C	D	E
X		1	1	-1	2	2
Y	5	1	-1	2	-2	-2

a) (2)

0	0	0	1	
0	0	0	1	
0	1	1	1	
0	0	0	1	

0	0	0	0	1	1	1	0	
1	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	1	

	1	1	-1	2	2	(1)
5	1	-1	2	-2	-2	

b) (2)

1	0	0	0	
1	0	0	0	
1	1	1	0	
1	0	0	0	

0	0	0	0	0	1	1	1	
0	0	0	1	1	1	1	1	
0	0	0	0	0	1	1	1	
0	0	0	1	0	1	1	1	

	1	1	-1	-2	-2	(1)
5	1	-1	2	3	3	

59. (continued)

c)

	1	1	1	0	0	0	0	0	0	0	0	1
	1	0	0	0	0	0	0	1	1	1	1	1
(2)	1	1	1	0	0	0	0	0	0	1	1	1
	0	0	0	0	0	0	0	1	1	1	1	1

	1	-1	-3	-2	-2
4	1	1	3	2	2

(1)

d)

	0	0	1	0	0	0	0	0	1	1	0	1
	1	0	1	1	0	0	0	0	1	1	0	1
(4)	0	0	1	0	0	1	0	0	1	1	1	1
	0	0	1	1	0	0	0	0	1	1	0	1

	1	-1	2	3	-2
4	1	1	-1	-2	2

(1)

60. (10)

	0	0	0	1	0	0	0	0	1	1	1	1
	1	0	0	0	0	0	0	0	1	0	0	0
(2)	0	0	0	1	0	0	0	0	1	1	1	1
	0	1	1	1	0	0	0	0	1	1	1	0

a)

b)

	1	0	1	1	0	0	0	0	0	1	0	1
	1	0	1	0	0	0	0	0	1	1	0	1
(4)	1	0	1	1	0	0	0	0	0	1	0	1
	0	0	1	0	0	1	0	0	1	1	1	1

c)

	1	1	1	1	0	0	0	0	0	0	0	1
	1	0	0	0	0	0	0	0	1	1	1	1
(2)	1	1	1	1	0	0	0	0	0	0	0	1
	1	1	1	0	0	0	0	0	0	1	1	1

d)

	0	0	0	0	0	0	0	1	1	1	1	1
	1	0	0	0	0	0	0	0	1	1	1	1
(2)	0	0	0	0	0	0	0	1	1	1	1	1
	1	1	1	0	0	0	0	0	0	1	1	1

A B C D E (1920)
3 1 1 (1 1)

	X	A	B	C	D	E
X		-1	-1	3	2	2
Y	4	2	1	-2	-1	-1

	-1	-1	3	2	2
4	2	1	-2	-1	-1

(1)

	-1	-1	-2	2	-3
4	2	1	2	-1	3

(1)

	-2	1	-2	-1	-1
5	4	-1	3	2	2

(1)

	2	1	2	-1	-1
5	-1	-1	-2	2	2

(1)

61. (8)

0	0	0	0	
1	1	1	1	
1	1	1	1	
0	0	0	0	

A B C D E (20)
(0 0 0) [0] [0]
(A ↔ $\overline{A}$, B ↔ $\overline{B}$)
(A ↔ $\overline{A}$, C ↔ $\overline{C}$)

	X	A	B	C	D	E
X		-1	-1	-1	0	0
Y	2	1	1	1	0	0

a) (8)

1	1	1	1	
1	1	1	1	
1	1	1	1	
0	0	0	0	

0	0	0	0		0	0	0	0
0	0	0	0		1	1	1	1
0	0	0	0		1	1	1	1
0	0	0	0		1	1	1	1

	-1	-1	-1	0	0	(1)
2	1	1	1	0	0	

62. (8)

0	0	0	0
0	1	1	0
0	1	1	0
1	0	0	1

A B C D E (160)
(2 2 2) (0 0)
(D ↔ $\overline{D}$, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		-1	-1	-1	2	2
Y	2	1	1	1	-1	-1

a) (4)

	1	1	1
0	1	1	1
0	1	1	1
0	0	0	1

0	0	0	0			0	0	0
0	0	0	0		1	1	1	0
0	0	0	0		1	1	1	0
1	0	0	0		1	1	1	1

	-1	-1	-1	2	2	(2)
2	1	1	1	-1	-1	

63. (8)

1	0	0	0
0	0	0	1
0	1	1	0
1	0	0	1

A B C D E (320)
(2 2) (0 0 0)

	X	A	B	C	D	E
X		1	1	-1	-1	-1
Y	5	-1	-1	2	2	2

a) (6)

1	0	1	0
1	0	1	1
0	0	1	0
1	0	1	1

0	0	0	0		1	1	0	1
0	0	0	0		0	1	0	1
0	1	0	0		1	1	1	1
0	0	0	0		1	1	0	1

	-1	-1	2	2	-3	(1)
3	1	1	-1	-1	2	

b) (2)

1	0	0	0
0	0	0	0
1	1	1	0
1	0	0	0

0	0	0	0		1	1	1	1
0	0	0	1		1	1	1	1
0	0	0	0		0	1	1	1
0	0	0	1		1	1	1	1

	1	1	-1	-1	-1	(1)
5	-1	-1	2	2	2	

64. (8)

0	0	0	0	
0	1	1	0	
1	1	1	0	
0	0	0	1	

A B C D E (960)
2 2 0 (0 0)
(A ↔ B, D ↔ $\bar{D}$, E ↔ $\bar{E}$)

	X	A	B	C	D	E
X		2	-1	2	1	1
Y	4	-1	2	-2	-1	-1

a) (4)

1	1	1	1	
0	1	1	1	
1	1	1	1	
0	0	0	1	

0	0	0	0	0	0	0	0	
0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	1	

	-2	-1	-2	1	1	
4	3	2	2	-1	-1	(1)

b) (4)

0	0	0	0	
0	1	1	1	
0	0	0	0	
0	0	0	1	

0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	0	
1	1	1	0	1	1	1	1	
0	0	0	0	1	1	1	1	

	2	-1	2	1	1	
4	-1	2	-2	-1	-1	(1)

65. (8)

0	0	0	1	
0	1	1	1	
1	1	1	0	
1	0	0	0	

A B C D E (80)
(0 0) (0 0 0)
(A ↔ $\bar{A}$, B ↔ $\bar{B}$)
(A ↔ $\bar{A}$, C ↔ $\bar{C}$, D ↔ $\bar{D}$, E ↔ $\bar{E}$)

	X	A	B	C	D	E
X		2	2	-1	-1	-1
Y	4	-2	-2	1	1	1

a) (8)

0	0	0	0	
0	0	0	0	
1	1	1	0	
1	0	0	0	

0	0	0	1	1	1	1	1	
0	1	1	1	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	

	2	2	-1	-1	-1	
4	-2	-2	1	1	1	(1)

66. (8)

0	0	0	0	
0	1	1	0	
1	0	0	1	
0	0	0	0	

A B C D E (240)
4 0 0 (0 0)
(B ↔ C, D ↔ $\bar{D}$)
(D ↔ $\bar{D}$, E ↔ $\bar{E}$)
(B ↔ $\bar{B}$, C ↔ $\bar{C}$, D ↔ $\bar{D}$)

	X	A	B	C	D	E
X		1	-1	1	2	2
Y	4	1	1	-1	-2	-2

a) (8)

0	0	0	1	
0	1	1	1	
0	0	0	1	
0	0	0	1	

0	0	0	0	1	1	1	0	
0	0	0	0	1	1	1	0	
1	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	0	

	1	-1	1	2	2	
4	1	1	-1	-2	-2	(1)

67. (6)

0	1	1	0
1	0	0	0
0	0	0	1
0	1	1	1

A B C D E (320)
(1 1) (1 1 1)

	X	A	B	C	D	E
X		-1	-1	1	1	1
Y	4	2	2	-1	-1	-1

a) (4)

1	1	1	0
1	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0	0	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	1

	2	-2	-1	-1	-1
3	-1	2	1	1	1

(1)

b) (2)

0	1	1	1
1	1	1	1
0	0	0	1
0	1	1	1

0	0	0	0	1	1	1	0
0	0	0	0	1	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	-1	-1	1	1	1
4	2	2	-1	-1	-1

(1)

68. (4)

0	0	0	1
0	1	1	0
0	1	1	0
1	0	0	0

A B C D E (80)
2 (0 0 0 0)
(B ↔ $\overline{B}$, C ↔ $\overline{C}$, D ↔ $\overline{D}$, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		-1	1	1	1	1
Y	3	2	-1	-1	-1	-1

a) (4)

0	0	0	1
0	1	1	1
0	1	1	1
1	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	1	0	0	0

	-1	1	1	1	1
3	2	-1	-1	-1	-1

(1)

69. (2)

0	0	0	1
0	1	1	0
0	1	1	0
1	0	0	1

A B C D E (32)
(1 1 1 1 1)

	X	A	B	C	D	E
X		-1	-1	-1	-1	-1
Y	2	1	1	1	1	1

a) (2)

1	1	1	1
1	1	1	0
1	1	1	0
1	0	0	0

0	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	1	1	1	1	1

	-1	-1	-1	-1	-1
2	1	1	1	1	1

(1)

67. (6)

0	1	1	0
1	0	0	0
0	0	0	1
0	1	1	1

A B C D E (320)
(1 1) (1 1 1)

	X	A	B	C	D	E
X		-1	-1	1	1	1
Y	4	2	2	-1	-1	-1

a) (4)

1	1	1	0
1	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0	0	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	1

	2	-2	-1	-1	-1
3	-1	2	1	1	1

(1)

b) (2)

0	1	1	1
1	1	1	1
0	0	0	1
0	1	1	1

0	0	0	0	1	1	1	0
0	0	0	0	1	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	-1	-1	1	1	1
4	2	2	-1	-1	-1

(1)

68. (4)

0	0	0	1
0	1	1	0
0	1	1	0
1	0	0	0

A B C D E (80)
2 (0 0 0 0)
(B ↔ $\overline{B}$, C ↔ $\overline{C}$, D ↔ $\overline{D}$, E ↔ $\overline{E}$)

	X	A	B	C	D	E
X		-1	1	1	1	1
Y	3	2	-1	-1	-1	-1

a) (4)

0	0	0	1
0	1	1	1
0	1	1	1
1	1	1	1

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	1	0	0	0

	-1	1	1	1	1
3	2	-1	-1	-1	-1

(1)

69. (2)

0	0	0	1
0	1	1	0
0	1	1	0
1	0	0	1

A B C D E (32)
(1 1 1 1 1)

	X	A	B	C	D	E
X		-1	-1	-1	-1	-1
Y	2	1	1	1	1	1

a) (2)

1	1	1	1
1	1	1	0
1	1	1	0
1	0	0	0

0	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	1	1	1	1	1

	-1	-1	-1	-1	-1
2	1	1	1	1	1

(1)

70.

0	0	0	0	
0	1	0	1	
0	1	1	0	
0	0	1	1	

A B C D E (384)
2 2 2 2 2
(A → B → C → D → E → A)
(B ↔ E, C ↔ D)

X

0	0	0	0	
0	0	0	0	
0	0	1	1	
0	0	1	1	

Y

0	0	0	0	
1	1	1	1	
1	1	1	0	
1	1	1	1	

Z

0	0	0	0	1	1	1	1
0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	X	Y	A	B	C	D	E
X			1	1	0	1	0
Y			-2	2	3	-1	-1
Z	2	3	1	-1	-1	0	1

71.

1	0	0	0	
0	1	0	0	
0	0	1	0	
0	1	1	1	

A B C D E (1920)
2 2 2 0 0
(B ↔ C, D ↔ E)

X

0	0	0	0	
0	0	0	0	
0	0	1	1	
0	0	1	1	

Y

1	0	0	0	
1	1	1	0	
1	0	1	0	
1	1	1	1	

Z

0	0	0	0	1	1	1	1
0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	1
0	1	0	1	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	X	Y	A	B	C	D	E
X			1	1	0	1	0
Y			-1	1	2	-1	-2
Z	2	4	1	-1	-1	0	2

72.

A	B	C	D	E	(1920)
3	3	1	1	1	

(A ↔ B, C ↔ D)

0	1	0	0	
0	0	0	0	
0	0	0	1	
1	0	1	1	

X

0	0	0	0	
0	0	0	0	
0	0	1	1	
0	0	1	1	

Y

0	1	0	0	
0	1	0	1	
0	1	0	1	
1	1	1	1	

Z

0	0	0	0	1	1	1	1	
0	0	0	0	1	0	1	0	
0	0	0	0	1	0	1	0	
0	0	0	0	1	0	1	0	
1	0	1	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	
0	0	0	0	1	1	1	1	

	X	Y	A	B	C	D	E
X			1	1	0	1	0
Y			-1	2	2	-1	3
Z	2	4	1	-1	-1	0	-2

73.

A	B	C	D	E	(480)
(2	2)	(2	2)	0	

(A ↔ C, B ↔ D)

0	1	0	0	
0	0	0	1	
0	0	1	0	
1	0	1	1	

X

0	0	0	0	
0	0	0	0	
0	0	1	1	
0	0	1	1	

Y

1	1	1	1	
1	0	1	1	
1	0	1	0	
1	0	1	0	

Z

0	0	0	0	0	1	0	0	
0	0	0	0	0	1	0	1	
0	0	0	0	0	1	0	1	
0	0	0	0	1	1	0	1	
0	0	0	0	1	1	0	1	
0	1	0	0	1	1	1	1	
0	1	0	0	1	1	1	1	
0	1	0	1	1	1	1	1	

	X	Y	A	B	C	D	E
X			1	1	0	1	0
Y			-2	-2	-1	1	-3
Z	2	3	1	1	1	-1	2

74.

	A	B	C	D	E (960)
	2	0	0	(0	0)

(B → C, D → $\bar{D}$, E → $\bar{E}$)

	0	1	1	0
	1	0	0	0
	0	0	0	1
	1	0	0	1

X

	0	0	0	0
	0	0	0	0
	0	0	1	1
	0	0	1	1

Y

	0	1	1	1
	0	0	0	1
	0	0	0	1
	0	0	0	1

Z

	0	0	0	0	1	1	1	0
	1	0	0	0	1	1	1	0
	0	0	0	0	1	1	1	0
	1	0	0	0	1	1	1	0
	1	0	0	0	1	1	1	1
	1	1	0	0	1	1	1	1
	1	0	0	0	1	1	1	1
	1	1	0	0	1	1	1	1

	X	Y	A	B	C	D	E
X			1	1	0	1	0
Y			1	-1	-1	2	2
Z	2	4	-1	0	1	-3	-2

75.

	A	B	C	D	E (1920)
	3	1	1	1	1

(B → C, D → E)

	0	0	0	1
	0	1	0	0
	0	0	1	0
	0	1	1	0

X

	0	0	0	0
	0	0	0	0
	0	0	1	1
	0	0	1	1

Y

	1	1	1	1
	1	1	1	0
	1	1	1	0
	1	1	1	0

Z

	0	0	0	0	0	0	0	1
	0	0	0	0	0	1	0	1
	0	0	0	0	0	0	0	1
	0	0	0	0	0	1	0	1
	0	0	0	0	0	1	1	1
	0	0	0	0	1	1	1	1
	0	0	0	0	0	1	1	1
	0	0	0	0	1	1	1	1

	X	Y	A	B	C	D	E
X			1	1	0	1	0
Y			-3	-1	-1	-2	-2
Z	2	4	3	0	1	1	2

76.

1	0	0	1
0	0	0	0
0	0	0	0
0	1	1	1

A B C D E (960)
3 (1 1) (1 1)

X

0	0	0	0
0	0	0	0
0	0	1	1
0	0	1	1

Y

1	0	1	1
1	1	1	1
0	0	0	0
1	1	1	1

Z

0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	0	1
0	1	0	0	1	1	1	1
0	0	0	0	1	1	1	1

	X	Y	A	B	C	D	E
X			1	1	0	1	0
Y			-2	-2	3	1	-1
Z	2	4	2	1	-2	-1	1

77.

0	0	0	0
0	0	1	1
1	1	1	0
0	1	0	0

A B C D E (960)
2 2 0 0 0
(A ↔ B, C ↔ $\overline{D}$)
(C ↔ D, E ↔ $\overline{E}$)

X

0	0	0	0
0	0	0	0
0	0	1	1
0	0	1	1

Y

0	0	0	0
1	1	1	1
1	1	1	0
1	1	1	1

Z

0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1
0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0

	X	Y	A	B	C	D	E
X			1	1	0	1	0
Y			-2	2	3	-1	-1
Z	-3	4	4	1	-2	2	1

78.

A	B	C	D	E (480)
1	(1	1)	(1	1)

(B ↔ D, C ↔ E)

1	0	0	1	
0	0	0	1	
0	0	0	1	
1	1	1	0	

X

0	0	0	0	
0	0	0	0	
0	0	0	0	
1	1	1	1	

Y

1	0	1	1	
0	0	1	1	
0	0	1	1	
0	0	1	0	

Z

0	0	0	0	1	1	0	1
0	0	0	0	1	1	0	1
0	0	0	0	1	1	0	1
0	0	0	0	1	1	0	1
1	1	0	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	0	0	1	1	1	1

	X	Y	A	B	C	D	E
X			1	1	1	0	0
Y			0	-1	-1	2	-1
Z	2	3	-1	0	0	-2	1

79.

A	B	C	D	E (960)
2	2	0	(0	0)

(A ↔ B, D ↔ $\bar{D}$, E ↔ $\bar{E}$)

0	0	0	0	
1	0	0	1	
1	1	1	0	
0	0	0	1	

X

0	0	0	0	
0	0	0	0	
0	0	0	0	
1	1	1	1	

Y

1	0	1	0	
1	0	1	1	
1	0	1	0	
1	0	1	1	

Z

0	0	0	0	0	1	0	1
0	0	0	0	1	1	0	1
0	1	0	0	1	1	1	1
0	1	0	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	1

	X	Y	A	B	C	D	E
X			1	1	1	0	0
Y			-1	0	1	1	-2
Z	-3	3	3	2	1	-1	2

80.

1	0	0	1
0	0	1	0
0	0	0	1
1	1	0	1

A B C D E (384)
1 1 1 1 1
(A → B → C → D → E → A)
(B ↔ E, C ↔ D)

X

0	0	0	0
0	0	0	0
0	0	0	0
1	1	1	1

Y

1	0	1	1
0	0	1	0
1	0	1	1
0	0	0	0

Z

0	0	0	0	1	1	0	1
0	0	0	0	1	1	1	1
0	0	0	0	0	1	0	1
0	0	0	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1
1	1	0	1	1	1	1	1

	X	Y	A	B	C	D	E
X			1	1	1	0	0
Y			1	-1	-3	2	-2
Z	4	4	-2	-1	1	-1	2

81.

0	0	0	0
1	0	1	1
1	1	0	1
0	0	0	1

A B C D E (1920)
1 1 1 1 1
(B ↔ C, D ↔ E)

X

0	0	0	0
0	0	0	0
0	0	0	0
1	1	1	1

Y

1	1	1	1
1	0	1	1
1	1	1	1
0	0	1	1

Z

0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	0	1
1	1	0	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1

	X	Y	A	B	C	D	E
X			1	1	1	0	0
Y			-3	-1	-2	2	-1
Z	-3	3	4	2	3	-1	1

82.

	0	1	1	0
	1	0	0	1
	1	0	0	1
	0	1	1	0

A	B	C	D	E (2)
(0	0	0	0	0)

$(A \leftrightarrow \overline{A},\ B \leftrightarrow \overline{B})$

Z

	0	0	0	0	1	1	1	0
	0	0	0	0	1	0	0	0
	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	0	0
	1	1	1	0	1	1	1	1
	1	0	0	0	1	1	1	1
	1	0	0	0	1	1	1	1
	0	0	0	0	1	1	1	0

X

0	0	0	0	
0	0	0	1	
0	0	0	1	
0	1	1	1	

Y

0	1	1	1	
1	1	1	1	
1	1	1	1	
1	1	1	1	

	X	Y	A	B	C	D	E
X			1	1	1	1	1
Y			-3	1	1	1	1
Z	2	2	1	-1	-1	-1	-1

83.

	0	0	1	0
	1	1	1	0
	0	1	1	1
	0	1	0	0

A	B	C	D	E (120)
0	0	0	0	0

$(B \rightarrow C \rightarrow D \rightarrow E \rightarrow \overline{B})$

$(\underline{B} \leftrightarrow \underline{E},\ C \leftrightarrow D)$

$(\overline{A} \leftrightarrow \overline{A},\ B \leftrightarrow \overline{B},\ D \leftrightarrow \overline{D})$

Z

	0	0	1	1	1	1	1	1
	0	0	1	1	1	1	1	1
	0	1	1	1	1	1	1	1
	0	0	1	1	1	1	1	1
	0	0	0	0	0	1	1	1
	0	0	0	0	0	0	1	1
	0	0	0	1	1	1	1	1
	0	0	0	0	0	1	1	1

X

0	0	0	1	
0	0	0	1	
0	0	0	1	
0	0	1	1	

Y

0	0	0	0	
1	1	0	0	
0	0	0	0	
0	1	0	0	

	X	Y	A	B	C	D	E
X			2	1	1	3	2
Y			3	-1	2	-2	1
Z	-3	4	-2	1	-1	3	1

TABLE II

8.

X

	A	B	C	D	E
a)	3	2	2	1	1
	2	1	2	1	1
	2	1	3	2	1
	2	1	3	1	2
	1	0	2	1	1
	1	-1	3	2	2
	1	-1	3	1	1
	0	-1	2	1	1
	-1	-2	3	2	1
	-1	-2	3	1	2
	-1	-2	2	1	1
	-2	-3	2	1	1
	3	2	1	-1	2
	2	1	1	-1	2
	2	1	1	-2	3
	2	1	2	-1	3
	1	0	1	-1	2
	1	-1	2	-2	3
	1	-1	1	-1	3
	0	-1	1	-1	2
	-1	-2	1	-2	3
	-1	-2	2	-1	3
	-1	-2	1	-1	2
	-2	-3	1	-1	2
	3	2	-1	-2	1
	2	1	-1	-2	1
	2	1	-2	-3	1
	2	1	-1	-3	2
	1	0	-1	-2	1
	1	-1	-2	-3	2
	1	-1	-1	-3	1
	0	-1	-1	-2	1
	-1	-2	-2	-3	1
	-1	-2	-1	-3	2

X

	A	B	C	D	E
a)	-1	-2	-1	-2	1
	-2	-3	-1	-2	1
	2	1	1	0	1
	3	1	2	1	2
	3	1	2	-1	2
	1	0	1	0	1
	2	-1	3	1	2
	2	-1	3	-1	2
	2	-1	2	1	3
	2	-1	2	-1	3
	1	-1	2	0	1
	1	-1	1	0	2
	1	-1	2	1	2
	1	-1	2	-1	2
	1	-2	3	1	2
	1	-2	3	-1	2
	1	-2	2	1	3
	1	-2	2	-1	3
	0	-1	1	0	1
	-1	-3	2	1	2
	-1	-3	2	-1	2
	-1	-2	1	0	1
	2	1	0	-1	1
	3	1	-1	-2	2
	3	1	1	-2	2
	1	0	0	-1	1
	2	-1	-1	-3	2
	2	-1	1	-3	2
	2	-1	-1	-2	3
	2	-1	1	-2	3
	1	-1	0	-2	1
	1	-1	0	-1	2
	1	-1	-1	-2	2
	1	-1	1	-2	2

X

	A	B	C	D	E
a)	1	-2	-1	-3	2
	1	-2	1	-3	2
	1	-2	-1	-2	3
	1	-2	1	-2	3
	0	-1	0	-1	1
	-1	-3	-1	-2	2
	-1	-3	1	-2	2
	-1	-2	0	-1	1
	3	1	1	-1	1
	2	0	1	-1	1
	3	-1	2	-2	1
	3	-1	1	-2	2
	3	-1	2	-1	2
	2	-1	1	-2	1
	2	-1	2	-1	1
	2	-1	1	-1	2
	2	-2	1	-3	1
	2	-2	3	-1	1
	2	-2	1	-1	3
	1	-1	1	-1	1
	1	-2	1	-2	1
	1	-2	2	-1	1
	1	-2	1	-1	2
	1	-3	2	-2	1
	1	-3	1	-2	2
	1	-3	2	-1	2
	0	-2	1	-1	1
	-1	-3	1	-1	1

Y

X	A	B	C	D	E
2	4	5	2	3	-1

9.

a)

X	A	B	C	D	E
Y	2	5	2	2	-1	-1

b)

X	A	B	C	D	E
Y	2	5	3	3	-1	1

a) X

A	B	C	D	E
3	2	2	1	1
2	2	1	1	1
2	3	1	2	1
1	1	1	1	1
1	2	0	1	1
1	2	1	2	1
1	1	1	2	2
1	3	-1	1	1
1	3	-1	2	2
1	3	1	2	2
1	2	2	3	1
1	2	1	3	2
0	2	-1	1	1
0	1	0	1	1
0	1	1	2	1
-1	3	-2	2	1
-1	3	-1	2	2
-1	2	-1	3	2
-1	2	1	3	2

a) X

A	B	C	D	E
-1	1	1	3	1
-1	2	-2	1	1
-1	2	-1	2	1
-1	1	-1	2	2
-1	0	1	2	1
-1	1	1	2	2
-2	2	-3	1	1
-1	1	-1	1	1
-2	2	-1	3	1
-2	1	-1	3	2
-2	1	1	3	2
-2	1	-2	1	1
-2	1	-1	2	1
-1	0	0	1	1
-3	1	-2	2	1
-3	1	-1	2	2
-3	-1	-1	2	2
-2	0	-1	1	1
-3	-1	-1	1	1

b) X

A	B	C	D	E
-3	-2	-2	1	-1
-2	-1	-2	1	-1
-2	-1	-3	2	-1
-1	-1	-1	1	-1
-1	0	-2	1	-1
-1	-1	-2	2	-1
-1	-1	-1	2	-2
-1	1	-3	1	-1
-1	1	-3	2	-2
-1	-1	-3	2	-2
-1	-2	-2	3	-1
-1	-1	-2	3	-2
0	1	-2	1	-1
0	0	-1	1	-1
0	-1	-1	2	-1
1	2	-3	2	-1
1	1	-3	2	-2
1	1	-2	3	-2
1	-1	-2	3	-2

b) X

A	B	C	D	E
1	-1	-1	3	-1
1	2	-2	1	-1
1	1	-2	2	-1
1	1	-1	2	-2
1	0	-1	2	-1
1	-1	-1	2	-2
2	3	-2	1	-1
1	1	-1	1	-1
2	1	-2	3	-1
2	1	-1	3	-2
2	-1	-1	3	-2
2	2	-1	1	-1
2	1	-1	2	-1
1	0	0	1	-1
3	2	-1	2	-1
3	1	-1	2	-2
3	1	1	2	-2
2	1	0	1	-1
3	1	1	1	-1

10.

a)

Y	X	A	B	C	D	E
	2	2	4	3	3	1

b)

Y	X	A	B	C	D	E
	2	2	2	3	3	-1

X

a)

A	B	C	D	E
3	1	2	2	-1
2	0	1	1	-1
3	-1	1	1	-1
3	-1	2	1	-2
2	-1	1	0	-1
2	-1	1	1	-2
3	-2	2	-1	-1
3	-2	1	-1	-2
3	-2	1	1	-2
2	-2	1	1	-3
2	-2	1	-1	-1
1	-1	0	0	-1
2	-3	1	-2	-1
2	-3	1	-1	-2
2	-3	-1	-1	-2
2	-2	-1	-1	-3
1	-2	0	-1	-1
1	-2	-1	-1	-2
1	-3	-1	-1	-1
1	-3	-1	-2	-2
0	-2	-1	-1	-1
-1	-3	-2	-2	-1

b)

A	B	C	D	E
2	2	1	1	1
3	2	1	1	2
1	1	0	0	1
2	2	1	-1	3
2	2	-1	-1	3
3	2	-1	-1	2
1	1	0	-1	2
2	1	-1	-1	2
2	2	-1	-1	1
1	1	-1	-1	3
2	1	-1	-2	3
1	1	-1	-1	1
1	0	-1	-1	2
1	1	-1	-2	2
1	1	-2	-2	1
1	1	-2	-2	3
1	-1	-2	-2	3
1	1	-2	-3	2
0	0	-1	-1	1
-1	-1	-2	-3	2
-1	-1	-2	-2	1

11.

a)

X	A	B	C	D	E
Y	3	5	3	-1	2	1

X

	A	B	C	D	E
a)	2	3	2	-1	1
	1	2	1	-1	0
	1	2	2	-1	1
	1	3	2	-2	1
	1	3	2	-2	-1
	1	2	3	-2	1
	1	2	3	-1	2
	0	1	1	-1	0
	0	1	2	-1	1
	-1	2	2	-3	1
	-1	2	2	-3	-1
	-1	2	3	-2	1
	-1	2	3	-2	-1
	-1	1	3	-2	2
	-1	1	3	-1	1
	-1	1	2	-1	0

X

	A	B	C	D	E
a)	-1	1	1	-2	0
	-1	1	2	-2	1
	-1	1	2	-2	-1
	-1	0	2	-1	1
	-2	1	2	-3	1
	-2	1	2	-3	-1
	-2	1	3	-2	1
	-2	1	3	-2	-1
	-2	-1	3	-2	1
	-2	-1	3	-1	2
	-1	0	1	-1	0
	-2	-1	2	-1	1
	-3	-1	2	-2	1
	-3	-1	2	-2	-1
	-3	-2	2	-1	1
	-2	-1	1	-1	0

b)

X	A	B	C	D	E
Y	3	4	5	2	2	1

X

	A	B	C	D	E
b)	3	1	-1	-1	1
	2	1	-1	-1	0
	2	0	-1	-1	1
	3	1	-2	-2	1
	3	1	-2	-2	-1
	3	-1	-2	-2	1
	3	-1	-2	-1	-2
	1	0	-1	-1	0
	2	-1	-1	-1	2
	2	-1	-2	-1	1
	2	-1	-3	-2	1
	2	-1	-3	-2	-1
	1	-1	-1	-1	1
	2	-2	-1	-1	3
	2	-2	-3	-1	1

X

	A	B	C	D	E
b)	1	-1	-2	-2	1
	1	-1	-2	-2	-1
	1	-1	-2	-1	0
	1	-2	-2	-1	1
	1	-2	-1	-1	2
	1	-2	-3	-2	1
	1	-2	-3	-2	-1
	1	-3	-2	-2	1
	1	-3	-2	-1	2
	0	-1	-1	-1	0
	0	-2	-1	-1	1
	-1	-3	-2	-2	1
	-1	-3	-2	-2	-1
	-1	-3	-1	-1	1
	-1	-2	-1	-1	0

c)

X	A	B	C	D	E
Y	3	3	4	-1	-1	1

X

	A	B	C	D	E
c)	2	1	1	1	0
	3	1	2	2	1
	3	1	2	2	-1
	1	0	1	1	0
	2	-1	3	2	1
	2	-1	3	2	-1
	1	-1	2	2	1
	1	-1	2	2	-1

X

	A	B	C	D	E
c)	1	-1	2	1	0
	1	-2	3	1	1
	1	-2	3	1	-1
	0	-1	1	1	0
	-1	-3	2	2	1
	-1	-3	2	2	-1
	-1	-2	1	1	0

d)

X	A	B	C	D	E
Y	2	4	3	1	1	2

X

	A	B	C	D	E
d)	2	3	-1	-1	-2
	1	2	-1	-1	-2
	1	2	-2	-1	-3
	0	1	-1	-1	-2
	-1	1	-2	-2	-3

X

	A	B	C	D	E
d)	-1	1	-1	-1	-3
	-1	0	-1	-1	-2
	-2	-1	-2	-1	-3
	-2	-1	-1	-1	-2
	-3	-2	-1	-1	-2

12.

a)

Y \ X	A	B	C	D	E
3	5	4	-1	2	2

a)

A	B	C	D	E
3	2	2	-1	-1
2	2	1	-1	-1
2	1	2	-1	-1
1	1	1	-1	-1
2	1	3	-1	-2
1	1	1	-2	-2
1	1	2	-1	-2
1	0	2	-1	-1
1	1	3	-2	-2
1	1	2	-2	-3
1	-1	3	-1	-1
1	-1	3	-2	-2
0	0	1	-1	-1
0	-1	2	-1	-1
-1	-1	2	-2	-3
-1	-1	3	-2	-2
-1	-2	3	-1	-2
-1	-1	1	-2	-2
-1	-1	2	-1	-2
-1	-2	2	-1	-1
-1	-1	1	-1	-1
-2	-3	2	-1	-1
-2	-2	1	-1	-1

b)

Y \ X	A	B	C	D	E
2	4	2	1	1	1

b)

A	B	C	D	E
1	3	-1	-1	-1
0	2	-1	-1	-1
-1	3	-1	-2	-2
-1	2	-1	-1	-2
-1	1	-1	-1	-1
-2	2	-1	-1	-3
-2	1	-1	-1	-2
-3	1	-1	-2	-2
-2	0	-1	-1	-1
-3	-1	-1	-1	-1

c)

Y \ X	A	B	C	D	E
3	4	3	-1	-1	-1

c)

A	B	C	D	E
2	2	1	1	1
1	1	1	1	1
1	1	2	2	1
1	1	3	2	2
0	0	1	1	1
-1	-1	3	2	2
-1	-1	2	2	1
-1	-1	1	1	1
-2	-2	1	1	1

13.

a)

	X	A	B	C	D	E
Y	3	5	3	-1	-1	2

X

a)

A	B	C	D	E
2	3	1	1	-2
1	2	1	1	-2
1	2	2	1	-3
0	1	1	1	-2
-1	1	1	1	-3
-1	1	2	2	-3
-1	0	1	1	-2
-2	-1	2	1	-3
-2	-1	1	1	-2
-3	-2	1	1	-2

b)

	X	A	B	C	D	E
Y	3	3	4	-1	-1	-1

X

b)

A	B	C	D	E
3	1	1	1	1
2	0	1	1	1
3	-1	2	2	1
2	-1	2	1	1
1	-1	1	1	1
2	-2	3	1	1
1	-2	2	1	1
1	-3	2	2	1
0	-2	1	1	1
-1	-3	1	1	1

14.

a)

	X	A	B	C	D	E
Y	3	3	5	2	1	1

X

a)

A	B	C	D	E
3	1	-1	-1	-1
2	0	-1	-1	-1
3	-1	-2	-1	-2
3	-1	-1	-2	-2
2	-1	-2	-1	-1
2	-1	-1	-1	-2
1	-1	-1	-1	-1
2	-2	-1	-1	-3
2	-2	-3	-1	-1
1	-2	-2	-1	-1
1	-2	-1	-1	-2
1	-3	-1	-2	-2
1	-3	-2	-1	-2
0	-2	-1	-1	-1
-1	-3	-1	-1	-1

b)

	X	A	B	C	D	E
Y	3	4	5	2	-1	2

X

b)

A	B	C	D	E
3	2	-1	1	-2
2	1	-1	1	-2
2	1	-2	1	-3
2	1	-1	2	-3
1	0	-1	1	-2
1	-1	-2	2	-3
1	-1	-1	1	-3
0	-1	-1	1	-2
-1	-2	-2	1	-3
-1	-2	-1	2	-3
-1	-2	-1	1	-2
-2	-3	-1	1	-2

c)

	X	A	B	C	D	E
Y	3	3	4	-1	-1	-1

X

c)

A	B	C	D	E
3	2	2	1	1
2	1	2	1	1
2	1	3	2	1
1	0	2	1	1
1	-1	3	2	2
1	-1	3	1	1
0	-1	2	1	1
-1	-2	3	2	1
-1	-2	2	1	1
-2	-3	2	1	1

15.

a)

	X	A	B	C	D	E
Y	2	3	2	-1	-1	0

X

a)

A	B	C	D	E
1	2	1	1	0
1	3	2	2	1
0	1	1	1	0
-1	2	3	2	1
-1	1	2	1	0
-1	1	2	2	1
-2	1	3	2	1
-1	0	1	1	0
-3	-1	2	2	1
-2	-1	1	1	0

16.

a)

	X	A	B	C	D	E
Y	2	3	3	1	1	1

X

a)

A	B	C	D	E
2	2	-1	-1	-1
1	1	-1	-1	-1
1	1	-1	-2	-2
1	1	-2	-2	-3
0	0	-1	-1	-1
-1	-1	-2	-2	-3
-1	-1	-1	-2	-2
-1	-1	-1	-1	-1
-2	-2	-1	-1	-1

17.

a)

	X	A	B	C	D	E
Y	4	5	-1	3	2	2

X

a)

A	B	C	D	E
1	3	-1	2	2
0	2	-1	1	1
-1	3	-1	1	1
-1	3	-2	2	1
-1	2	-1	1	0
-1	2	-2	1	1
-2	3	-2	1	-1
-2	3	-2	1	1
-2	2	-3	1	1
-1	1	-1	0	0
-3	2	-2	-1	-1

b)

	X	A	B	C	D	E
Y	3	3	2	2	1	4

X

b)

A	B	C	D	E
2	-1	-1	3	-2
1	-1	-1	2	-2
1	-2	-1	2	-3
0	-1	-1	1	-2
-1	-2	-2	1	-3
-1	-1	-1	1	-3
-1	-1	-1	0	-2

c)

	X	A	B	C	D	E
Y	4	1	3	3	2	2

X

c)

A	B	C	D	E
3	-1	-1	1	1
2	-1	-1	1	0
3	-2	-2	1	1
3	-2	-2	1	-1
1	-1	-1	0	0
2	-2	-3	-1	-1
1	-2	-2	-1	-1

d)

	X	A	B	C	D	E
Y	5	2	-1	-1	3	3

X

d)

A	B	C	D	E
3	2	2	1	1
1	1	1	0	0
2	3	2	-1	-1
1	2	2	-1	-1

e)

	X	A	B	C	D	E
Y	2	3	1	1	2	2

X

e)

A	B	C	D	E
-3	-1	-1	-2	-2

18.

a)

Y	X	A	B	C	D	E
	4	2	1	3	3	2

b)

Y	X	A	B	C	D	E
	5	3	2	-1	4	-2

a)

X	A	B	C	D	E
	3	2	-1	-1	-2
	2	2	-1	-1	-1
	2	1	-1	-1	-2
	1	1	-1	-1	-1
	2	1	-1	-2	-3
	1	1	-1	-2	-2
	1	0	-1	-1	-2
	1	1	-2	-2	-1
	1	1	-2	-2	-3
	1	1	-2	-3	-2
	1	-1	-2	-2	-3
	0	0	-1	-1	-1

b)

X	A	B	C	D	E
	2	1	3	-1	2
	1	1	2	-1	2
	1	0	2	-1	1
	1	1	3	-2	2
	1	1	2	-2	3
	1	-1	3	-2	2
	0	0	1	-1	1
	-1	-1	2	-3	2

c)

Y	X	A	B	C	D	E
	5	3	2	-1	-1	3

d)

Y	X	A	B	C	D	E
	3	4	1	2	2	-1

e)

Y	X	A	B	C	D	E
	3	4	3	2	2	1

c)

X	A	B	C	D	E
	1	1	2	2	-1
	1	1	3	2	-2
	0	0	1	1	-1
	-1	-1	2	2	-3

d)

X	A	B	C	D	E
	-2	2	-1	-1	3
	-2	1	-1	-1	2
	-3	1	-1	-2	2
	-2	0	-1	-1	1

e)

X	A	B	C	D	E
	-2	-3	-1	-1	-2
	-2	-2	-1	-1	-1

19.

	A	B	C	D	E	X	A	B	C	D	E
a)	1	3	-1	-1	-1	2	1	-1	1	1	1
	0	2	-1	-1	-1	2	1	-1	1	1	1
	-1	3	-1	-2	-2	3	2	-1	2	2	1
	-1	2	-1	-1	-2	3	3	-1	2	2	2
	-1	1	-1	-1	-1	3	3	-1	2	2	2
b)	-3	-1	-1	-1	-1	2	3	1	1	1	1
c)	1	1	1	1	1	4	1	-1	-1	-1	-1

20.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	3	2	-2	-1	3	2	0	1	2	1
	-1	2	1	-2	-1	3	2	0	1	2	1
	-2	2	1	-3	-1	3	2	0	1	2	1
	-1	2	1	-3	-2	4	3	1	2	3	2
	-1	1	0	-2	-1	4	3	1	2	3	2
	-2	1	-1	-3	-2	4	3	1	2	3	2
b)	-1	-2	-3	-2	-1	3	2	3	4	2	1
c)	-1	2	-2	-1	3	3	2	1	3	1	-1
	-1	1	-2	-1	2	3	2	1	3	1	-1
	-2	1	-3	-1	2	3	2	1	3	1	-1
	-1	1	-3	-2	2	4	3	2	5	2	-1
	-1	0	-2	-1	1	4	3	2	5	2	-1
d)	-1	2	1	2	3	4	3	1	2	-1	-2
	-1	1	0	1	2	4	3	1	2	-1	-2
	-2	1	-1	2	3	4	3	1	2	-1	-2
e)	3	2	1	2	-1	3	-1	0	1	-1	1
	2	1	0	1	-1	4	-1	1	2	-1	2
	3	1	-1	2	-2	4	-1	1	2	-1	2

21.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	3	-1	-1	-1	4	3	-3	1	1	1
b)	-1	3	1	-2	-2	5	4	-3	-1	2	2
c)	3	1	1	1	1	4	-1	-1	-1	-1	-1

22.

	A	B	C	D	E	X	A	B	C	D	E
a)	1	2	2	1	1	4	1	-1	-1	-1	-1
b)	3	-1	-1	2	2	3	0	2	2	-1	-1
	2	-1	-1	2	1	3	0	2	2	-1	-1
	1	-1	-1	1	1	3	0	2	2	-1	-1
	1	-2	-2	1	1	4	1	3	3	-1	-1
c)	1	-1	-1	3	-1	3	1	2	2	-2	1
	0	-1	-1	2	-1	3	1	2	2	-2	1
	-1	-2	-2	3	-1	4	2	3	3	-2	1
d)	-2	-1	-1	2	-3	3	3	2	2	-1	2
	-2	-1	-1	1	-2	3	3	2	2	-1	2
e)	-3	-1	-1	-1	-1	3	4	2	2	1	1
f)	1	3	-1	-1	-1	3	1	-1	2	1	1
	0	2	-1	-1	-1	3	1	-1	2	1	1
	-1	3	-2	-1	-2	4	2	-1	3	1	2
	-1	2	-2	-1	-1	5	3	-1	4	2	2

23.

	A	B	C	D	E	X	A	B	C	D	E
a)	3	-1	-1	1	1	2	-1	1	1	0	0
	2	-1	-1	1	0	3	-1	2	2	0	1
	3	-2	-2	1	-1	3	-1	2	2	0	1
	3	-2	-2	1	1	4	-1	3	3	1	1
	1	-1	-1	0	0	4	-1	3	3	1	1
b)	-1	-1	-1	1	-3	2	1	1	1	0	2
	-1	-1	-1	0	-2	3	2	2	2	1	3
c)	1	1	1	0	0	4	-1	-1	-1	1	1

24.

	A	B	C	D	E	X	A	B	C	D	E
a)	2	2	-1	-1	-1	2	0	0	1	1	1
	1	1	-1	-1	-1	2	0	0	1	1	1
	1	1	-1	-2	-2	3	1	1	2	2	2
	1	1	-2	-2	-3	3	1	1	2	2	2
	0	0	-1	-1	-1	3	1	1	2	2	2
b)	1	1	2	2	-1	3	1	1	-1	-1	2
	1	1	3	2	-2	3	1	1	-1	-1	2
	0	0	1	1	-1	3	1	1	-1	-1	2
	-1	-1	2	2	-3	4	2	2	-1	-1	3
c)	-2	-2	-1	-1	-1	2	2	2	1	1	1

25.

	A	B	C	D	E	X	A	B	C	D	E
a)	1	3	-1	2	2	4	2	-2	1	-1	-1
	0	2	-1	1	1	5	3	-3	2	-1	-1
b)	1	1	1	2	2	5	2	-1	-1	-2	-2
c)	3	2	2	-1	-1	3	0	-1	-1	1	1
	2	2	1	-1	-1	4	1	-1	-1	2	2
	1	1	1	-1	-1	4	1	-1	-1	2	2
	1	1	1	-2	-2	5	2	-1	-1	3	3
d)	-2	2	-3	-1	-1	3	3	-1	2	1	1
	-2	1	-2	-1	-1	4	5	-1	3	2	2
e)	-3	-1	-1	-2	-2	3	4	1	1	2	2
f)	2	-1	-1	3	-2	3	1	1	1	-1	2
	1	-1	-1	2	-2	3	1	1	1	-1	2
	1	-1	-2	2	-3	4	2	1	2	-1	3
	0	-1	-1	1	-2	5	3	2	2	-1	4

26.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	2	1	3	-2	2	1	0	0	-1	1
	-1	1	0	2	-1	4	3	1	1	-2	2
	-2	1	-1	3	-2	4	3	1	1	-2	2
	-1	1	1	2	-2	3	2	1	0	-1	2
	-2	1	1	3	-2	5	4	2	1	-2	3
	-2	1	1	2	-3	3	2	1	0	-1	2
	-1	0	0	1	-1	5	4	2	1	-2	3
b)	2	1	0	-1	-1	4	-1	1	1	2	2
	3	1	-1	-2	-2	4	-1	1	1	2	2
	3	1	1	-2	-2	5	-1	2	1	3	3
	1	0	0	-1	-1	5	-1	2	1	3	3
c)	-1	-3	1	-2	-2	2	1	2	0	1	1
	-1	-2	0	-1	-1	4	3	5	1	2	2
d)	3	1	1	2	2	3	-1	1	0	-1	-1
	1	0	0	1	1	5	-1	2	1	-2	-2

27.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	2	1	2	-3	3	2	0	1	-1	2
	-1	1	0	1	-2	3	2	0	1	-1	2
	-2	1	-1	2	-3	5	4	1	2	-2	3
b)	-1	-3	1	2	2	3	2	3	1	-1	-1
	-1	-2	0	1	1	3	2	3	1	-1	-1
c)	2	1	0	1	1	3	-1	0	1	-1	-1
	3	1	-1	2	2	5	-1	1	2	-2	-2

28.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	-1	1	3	4	3	2	2	1	-3
	-1	-1	-1	0	2	4	3	2	2	1	-3
b)	-1	3	-2	2	1	3	2	-1	2	0	-1
	-1	2	-2	1	1	3	2	-1	2	0	-1
	-2	2	-3	1	1	4	3	-1	3	1	-1
c)	-1	3	-1	1	-1	4	3	-2	2	1	1
	-1	2	-1	0	-1	4	3	-2	2	1	1
	-2	3	-2	-1	-1	5	4	-2	3	2	1
d)	-1	-2	-2	-3	1	3	2	2	2	3	-1
e)	-1	-1	-1	-3	-1	4	3	2	2	5	1
f)	3	-1	-1	1	-1	4	-1	2	2	1	1
	2	-1	-1	0	-1	4	-1	2	2	1	1
	3	-2	-2	-1	-1	5	-1	3	3	2	1
g)	3	2	2	1	1	4	-1	-1	-1	1	-1

29.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-2	-2	-1	-3	4	3	2	2	1	3
b)	-1	2	-1	3	-2	4	3	-1	1	-2	2
	-1	1	-1	2	-2	5	4	-1	2	-2	3

30.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	-1	-1	-3	3	2	1	1	1	3
b)	-1	2	2	-3	1	3	2	-1	-1	2	0
	-1	1	1	-2	0	4	3	-1	-1	3	1
c)	-1	3	-1	-1	1	3	2	-2	1	1	0
	-1	2	-1	-1	0	5	4	-3	2	2	1
d)	3	-1	-1	-1	1	3	-1	1	1	1	0
	2	-1	-1	-1	0	5	-1	2	2	2	1
e)	2	1	1	1	0	4	-1	-1	-1	-1	1

31.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	3	2	2	3	2	1	-1	-1	-1
	-1	-1	2	2	1	4	3	2	-1	-2	-1
	-1	-1	1	1	1	5	4	3	-1	-2	-2
b)	-1	-2	-1	3	-2	3	2	2	1	-2	1
	-2	-2	-1	3	-1	4	3	3	2	-2	1

32.

	A	B	C	D	E	X	A	B	C	D	E
a)	2	-2	-1	-3	1	4	1	3	2	2	-1
	1	-2	-1	-2	1	4	1	3	2	2	-1
	1	-3	-2	-2	1	5	2	4	3	2	-1
b)	-3	-2	-1	2	1	4	5	3	2	-2	-1
c)	1	2	3	-2	1	5	2	-1	-2	2	-1
d)	2	3	-1	2	1	4	1	-1	2	-2	-1
	1	2	-1	2	1	4	1	-1	2	-2	-1
	1	2	-2	3	1	5	2	-1	3	-3	-1

33.

	A	B	C	D	E	X	A	B	C	D	E
a)	1	2	-1	-1	0	2	0	-1	1	1	0
	1	3	-2	-2	1	3	1	-1	2	2	0
	0	1	-1	-1	0	3	1	-1	2	2	0
b)	-2	-1	-1	-1	0	2	2	1	1	1	0
c)	0	1	1	1	0	3	1	-1	-1	-1	0

34.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	1	1	2	-2	2	1	0	0	-1	1
	-2	1	1	3	-2	4	3	1	1	-2	2
	-1	0	0	1	-1	4	3	1	1	-2	2
b)	3	1	1	2	2	4	-1	1	1	-2	-2
	1	0	0	1	1	4	-1	1	1	-2	-2

35.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	1	3	2	2	4	2	1	-2	-1	-1
	-1	0	2	1	1	5	3	2	-3	-1	-1
b)	-2	-3	2	-1	-1	3	2	3	-1	1	1
c)	-2	2	-3	-1	-1	3	2	0	2	1	1
	-2	1	-2	-1	-1	5	4	1	3	2	2
d)	-1	1	-2	2	-3	4	2	1	2	-1	3
	-1	0	-1	1	-2	5	3	2	2	-1	4
e)	2	0	-1	1	1	5	-2	2	2	-1	-1
f)	3	2	2	-1	-1	3	-1	0	-1	1	1
	2	1	2	-1	-1	5	-1	1	-2	2	2

36.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	3	-2	-2	2	1	1	-1	1	1
	-1	-1	2	-1	-2	3	2	2	-1	1	2
	-1	-1	1	-1	-1	4	3	3	-1	2	2

37.

	A	B	C	D	E	X	A	B	C	D	E
a)	0	2	-1	-1	-1	4	2	-2	1	1	1
b)	1	-1	3	-2	-2	3	1	1	-2	1	1
	0	-1	2	-1	-1	4	2	2	-3	1	1
c)	2	2	1	1	1	3	0	-1	-1	-1	-1
	1	1	1	1	1	5	1	-1	-2	-2	-2
d)	-2	-2	1	1	1	3	3	2	-1	-1	-1

38.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-2	3	2	-1	4	2	3	-2	-1	1
	-1	-2	2	1	-1	5	3	4	-2	-1	2
b)	3	-1	-2	-1	2	4	-1	2	2	1	-1
	2	-1	-2	-1	1	5	-1	3	3	2	-1
c)	-1	-2	-2	-3	-1	4	2	3	2	3	1

39.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	1	1	-2	0	3	2	-1	-1	2	0
b)	2	1	1	1	0	3	-1	-1	-1	-1	0

40.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	2	2	1	4	2	2	-1	-1	-1
	-1	-1	1	1	1	5	3	3	-1	-1	-2
b)	-2	-2	-1	-1	3	3	2	2	1	1	-2
c)	-1	-1	-2	-2	1	4	2	2	3	3	-1
d)	3	-2	-1	-1	-2	3	-1	2	1	1	1
	2	-2	-1	-1	-1	4	-1	3	2	2	1

41.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	-1	3	-1	5	3	2	2	-4	1
b)	-2	2	-3	1	1	4	3	-1	3	-1	-1
c)	-1	3	-1	-1	-1	5	3	-3	2	1	1
d)	3	2	2	1	1	4	-1	-1	-1	-1	-1
e)	3	-1	-1	-1	-1	5	-2	2	2	1	1

42.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	2	2	1	1	4	3	-2	-2	-1	-1

43.

	A	B	C	D	E	X	A	B	C	D	E
a)	3	-2	-2	1	1	2	-1	1	1	0	0
	1	-1	-1	0	0	5	-2	3	3	1	1
b)	1	1	1	0	0	5	-2	-2	-2	1	1

44.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-2	0	-1	-1	5	3	4	1	2	2
b)	-1	1	0	2	-1	5	3	-1	1	-3	2
c)	-2	-1	1	2	-3	3	2	1	0	-1	2
d)	3	-1	1	2	2	3	-1	1	0	-1	-1
e)	2	1	0	-1	-1	5	-2	-1	1	2	2

45.

	A	B	C	D	E	X	A	B	C	D	E
a)	-2	-2	1	1	1	4	3	3	-1	-1	-1
b)	-1	-1	3	-1	-1	4	2	2	-3	1	1
c)	3	-1	-1	-1	-1	4	-2	2	1	1	1
d)	2	2	1	1	1	4	-1	-1	-1	-1	-1

46.

	A	B	C	D	E	X	A	B	C	D	E
a)	1	-2	-2	1	1	3	0	2	2	-1	-1
b)	0	-1	-1	2	-1	4	1	2	2	-3	1
c)	1	2	2	1	1	3	0	-1	-1	-1	-1
d)	0	2	-1	-1	-1	4	1	-2	2	1	1

47.

	A	B	C	D	E	X	A	B	C	D	E
a)	3	-1	-1	-1	-1	3	-2	1	1	1	1
b)	1	1	1	1	1	4	-1	-1	-1	-1	-1

48.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	2	2	-1	-1	3	2	-1	-1	1	1
	-1	1	1	-1	-1	4	3	-1	-1	2	2
b)	-1	-1	-1	-2	-2	3	2	1	1	2	2

49.

	A	B	C	D	E	X	A	B	C	D	E
a)	-2	2	-3	1	1	5	3	-2	3	-1	-1
b)	3	2	2	1	1	5	-2	-2	-2	-1	-1

50.

	A	B	C	D	E	X	A	B	C	D	E
a)	-2	1	1	2	-3	5	3	1	-1	-2	3
b)	3	1	1	2	2	5	-2	1	-1	-2	-2

51.

	A	B	C	D	E	X	A	B	C	D	E
a)	-2	2	-3	-1	-1	5	3	-1	4	2	2
b)	3	2	2	-1	-1	5	-2	-1	-1	2	2

52.

	A	B	C	D	E	X	A	B	C	D	E
a)	-2	-3	-1	2	1	5	3	4	2	-2	-1
b)	-2	2	-1	-3	1	5	3	-1	2	3	-1
c)	3	2	-1	2	1	5	-2	-1	2	-2	-1

53.

	A	B	C	D	E	X	A	B	C	D	E
a)	-2	1	-1	-3	2	5	3	1	2	4	-2
b)	-2	1	2	-1	3	5	3	1	-1	2	-3
c)	3	1	-1	2	2	5	-2	1	2	-1	-2

54.

	A	B	C	D	E	X	A	B	C	D	E
a)	-2	-1	3	-2	1	5	4	2	-3	2	-1
b)	-1	2	2	-3	-1	5	3	-1	-2	3	1
c)	-1	-3	2	2	-1	5	3	4	-2	-2	1
d)	3	-1	-2	-2	1	5	-1	2	2	2	-1

55.

	A	B	C	D	E	X	A	B	C	D	E
a)	3	-2	-2	1	-1	5	-2	3	3	-1	1

56.

	A	B	C	D	E	X	A	B	C	D	E
a)	0	2	-1	-1	-1	3	1	-2	1	1	1
b)	1	1	1	1	1	3	0	-1	-1	-1	-1

57.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	2	-2	1	1	5	3	-2	3	-1	-1
b)	-2	3	-2	-1	-1	4	3	-2	2	1	1
c)	3	-2	-2	-1	-1	4	-1	2	2	1	1

58.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	1	1	-2	-2	5	2	-1	-1	3	3
b)	-3	-1	-1	-2	-2	4	3	1	1	2	2
c)	-1	1	1	2	2	5	2	-1	-1	-2	-2
d)	2	-1	-1	3	-2	4	-1	1	1	-2	2

59.

	A	B	C	D	E	X	A	B	C	D	E
a)	1	1	-1	2	2	5	1	-1	2	-2	-2
b)	1	1	-1	-2	-2	5	1	-1	2	3	3
c)	1	-1	-3	-2	-2	4	1	1	3	2	2
d)	1	-1	2	3	-2	4	1	1	-1	-2	2

60.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	3	2	2	4	2	1	-2	-1	-1
b)	-1	-1	-2	2	-3	4	2	1	2	-1	3
c)	-2	1	-2	-1	-1	5	4	-1	3	2	2
d)	2	1	2	-1	-1	5	-1	-1	-2	2	2

61.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	-1	0	0	2	1	1	1	0	0

62.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	-1	2	2	2	1	1	1	-1	-1
	-1	-1	-1	1	1	3	2	2	2	-1	-1

63.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	2	2	-3	3	1	1	-1	-1	2
b)	1	1	-1	-1	-1	5	-1	-1	2	2	2

64.

	A	B	C	D	E	X	A	B	C	D	E
a)	-2	-1	-2	1	1	4	3	2	2	-1	-1
b)	2	-1	2	1	1	4	-1	2	-2	-1	-1

65.

	A	B	C	D	E	X	A	B	C	D	E
a)	2	2	-1	-1	-1	4	-2	-2	1	1	1

66.

	A	B	C	D	E	X	A	B	C	D	E
a)	1	-1	1	2	2	4	1	1	-1	-2	-2

67.

	A	B	C	D	E	X	A	B	C	D	E
a)	2	-2	-1	-1	-1	3	-1	2	1	1	1
b)	-1	-1	1	1	1	4	2	2	-1	-1	-1

68.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	1	1	1	1	3	2	-1	-1	-1	-1

69.

	A	B	C	D	E	X	A	B	C	D	E
a)	-1	-1	-1	-1	-1	2	1	1	1	1	1

Ford Motor Company
AERONUTRONIC DIVISION

FUNCTION INDEX

A	B	C	D	E	Fcn. No.
8	0	0	0	0	1
7	1	1	1	1	6
6	2	2	2	0	4
6	2	2	0	0	9
6	2	0	0	0	13
6	0	0	0	0	21
5	3	3	1	1	7
5	3	(1	1	1)	12
5	3	(1	1)	1	11
5	(1	1)	(1	1)	25,29
5	(1	1	1)	1	30
4	4	4	0	0	2
4	4	2	2	2	5
4	4	2	2	0	8
4	4	2	0	0	14
4	4	(0	0	0)	16
4	4	(0	0)	[0]	15
4	(2	2	2	2)	19
4	(2	2)	(2	2)	17
4	2	2	2	0	18,28
4	(2	2)	(0	0)	41
4	2	2	(0	0)	27,31
4	2	2	0	0	32
4	2	(0	0	0)	37
4	2	(0	0)	0	54
4	(0	0	0)	[0]	39
4	(0	0)	(0	0)	42
4	0	0	(0	0)	66
3	3	3	3	3	3
3	3	3	3	1	10
(3	3	3)	(1	1)	23
3	(3	3)	(1	1)	22
3	3	3	1	1	20
(3	3)	(1	1	1)	24,36,45
3	3	1	(1	1)	26,35
3	3	1	1	1	38,72
3	(1	1	1	1)	56
3	(1	1)	(1	1)	48,57,76
3	1	1	(1	1)	44,59,60
3	1	1	1	1	75
(2	2	2	2	2)	47
2	(2	2)	(2	2)	51
2	2	2	2	2	70
2	(2	2	2)	[0]	33
(2	2)	(2	2)	0	40,73
2	2	2	2	0	53
(2	2	2)	(0	0)	55,62
2	(2	2)	(0	0)	34,46
2	2	2	0	0	52,71
(2	2)	(0	0	0)	63
2	2	0	(0	0)	50,64,79
2	2	0	0	0	77
2	(0	0	0	0)	68
2	(0	0)	(0	0)	49
2	0	0	(0	0)	74
(1	1	1	1	1)	69
(1	1	1)	(1	1)	43,67
1	(1	1)	(1	1)	58,78
1	1	1	1	1	80,81
(0	0	0)	[0]	[0]	61
(0	0	0	0	0)	82
(0	0)	(0	0	0)	65
0	0	0	0	0	83

DISTRIBUTION

Assistant Secretary of Defense
for Research and Eng
Pentagon Building
Washington 25, D. C. — 2 Copies

Armed Services Technical
Information Agency
Arlington Hall Station
Arlington 12, Virginia — 10 Copies

Chief of Naval Research
Department of the Navy
Washington 25, D. C.

Attn: Code 437,
Information Systems Branch
2 Copies

Chief of Naval Operations
OP-07T-12
Navy Department
Washington 25, D. C. — 1 Copy

Director
Naval Research Laboratory
Washington 25, D. C.

Attn: Technical Information Officer
Code 2000 — 6 Copies

Commanding Officer
Office of Naval Research
Navy No. 100, Fleet Post Office
New York, New York
10 Copies

Commanding Officer
ONR Branch Office
346 Broadway
New York 13, New York
1 Copy

Commanding Officer
ONR Branch Office
495 Summer Street
Boston 10, Massachusetts
1 Copy

Bureau of Ships
Department of the Navy
Washington 25, D. C.

Attn: Code 607A NTDS — 1 Copy

Bureau of Naval Weapons
Department of the Navy
Washington 25, D. C.

Attn: RAAV Avionics Division
1 Copy

Bureau of Naval Weapons
Department of the Navy
Washington 25, D. C.

Attn: RMWC Missile Weapons
Control Div — 1 Copy

Bureau of Naval Weapons
Department of the Navy
Washington 25, D. C.

Attn: RUDC ASW Detection
and Control Div — 1 Copy

DISTRIBUTION (Continued)

Bureau of Ships
Department of the Navy
Washington 25, D. C.

Attn: Communications Branch
Code 686 — 1 Copy

Naval Ordnance Laboratory
White Oaks
Silver Spring 19, Maryland

Attn: Technical Library — 1 Copy

David Taylor Model Basin
Washington 7, D. C.

Attn: Technical Library — 1 Copy

Naval Electronics Laboratory
San Diego 52, California

Attn: Technical Library — 1 Copy

University of Illinois
Control Systems Laboratory
Urbana, Illinois

Attn: D. Alpert — 1 Copy

Air Force Cambridge Research Laboratories
Laurence C. Hanscom Field
Bedford, Massachusetts

Attn: Research Library, CRX2-R — 1 Copy

Technical Information Officer
U. S. Army Signal Research and Development Lab
Fort Monmouth, New Jersey

Attn: Data Equipment Branch — 1 Copy

National Security Agency
Fort George G. Meade,
Maryland

Attn: R-4, Howard Campaigne — 1 Copy

U. S. Naval Weapons Laboratory
Dahlgren, Virginia

Attn: Head, Compution Div,
G. H. Gleissner — 1 Copy

National Bureau of Standards
Data Processing Systems Div
Room 239, Bldg 10
Washington 25, D. C.

Attn: A. K. Smilow — 1 Copy

Aberdeen Proving Ground, BRL
Aberdeen Proving Ground, Maryland

Attn: J. H. Giese,
Chief Compution Lab — 1 Copy

Commanding Officer
ONR Branch Office
John Crerar Library Bldg
86 East Randolph Street
Chicago 1, Illinois — 1 Copy

DISTRIBUTION (Continued)

Commanding Officer
ONR Branch Office
1030 E. Green Street
Pasadena, California — 1 Copy

Commanding Officer
ONR Branch Office
1000 Geary Street
San Francisco 9, California — 1 Copy

National Bureau of Standards
Washington 25, D. C.

Attn: Mr. R. D. Elbourn — 1 Copy

George Washington University
Washington, D. C.

Attn: Prof. N. Grisamore — 1 Copy

Syracuse University
Electrical Eng Dpt
Syracuse 10, New York

Attn: Dr. Stanford Goldman — 1 Copy

Burroughs Corporation
Research Center
Paoli, Pennsylvania

Attn: R. A. Tracy — 1 Copy

Cornell University
Cognitive Systems Research Program
Hollister Hall
Ithaca, New York

Attn: Dr. Frank Rosenblatt — 1 Copy

Lockheed Missiles and Space Company
3251 Hanover Street
Palo Alto, California

Attn: W. F. Main — 1 Copy

Communications Sciences Lab
University of Michigan
180 Frieze Building
Ann Arbor, Michigan

Attn: Gordon E. Peterson — 1 Copy

University of Michigan
Ann Arbor, Michigan

Attn: Dept of Psychology,
Prof. Arthur Melton — 1 Copy

Carnegie Institute of Technology
Dept of Psychology
Pittsburgh 13, Pennsylvania

Attn: Prof. Bert F. Green, Jr — 1 Copy

Stanford University
Stanford, California

Attn: Electronic Lab
Prof. Gene Franklin — 1 Copy

DISTRIBUTION (Continued)

University of Illinois
Urbana, Illinois

Attn: Electrical Engrg Dept
Prof. H. Von Foerster
1 Copy

University of California
Institute of Eng Research
Berkeley 4, California

Attn: Prof. A. J. Thomasian
1 Copy

University of California - LA
Los Angeles 24, California

Attn: Dept of Engineering,
Prof. Gerald Estrin
1 Copy

University of Illinois
Champaign Urbana, Illinois

Attn: John R. Pasta 1 Copy

Naval Research Laboratory
Washington 25, D. C.

Attn: Security Systems
Code 5266, Mr. G. Abraham
1 Copy

Zator Company
140 1/2 Mt. Auburn
Cambridge 38, Massachusetts

Attn: R. J. Solomonoff
1 Copy

Telecomputing Corporation
12838 Saticoy Street
North Hollywood, California

Attn: Data Instruments Div
Field Engineering Dept
1 Copy

NASA
Goddard Space Flight Center
Washington 25, D. C.

Attn: Arthur Shapiro 1 Copy

Dr. W. Ross Ashby
University of Illinois
Dept of Electrical Engineering
Urbana, Illinois 1 Copy

National Physical Laboratory
Teddington, Middlesex
England

Attn: Dr. A. M. Uttley, Supt
Autonomics Division
1 Copy

University College London
Department of Elec. Eng.
Gower Street
London, W. C. 1, England

Attn: Dr. W. K. Taylor

Dr. George B. Yntema
United Aircraft Corporation
Research Laboratories
East Hartford 8, Connecticut
1 Copy

DISTRIBUTION (Continued)

Swarthmore College
Swarthmore, Pennsylvania

Attn: Department of Electrical Engrg
Prof. Carl Barus
1 Copy

Dr. Jacob Beck
Harvard University
Memorial Hall
Cambridge 38, Massachusetts
1 Copy

Diamond Ordnance Fuze laboratory
Connecticut Ave and Van Ness St
Washington 25, D. C.

Attn: ORDTL-012,
E. W. Channel 1 Copy

Harvard University
Cambridge, Massachusetts

Attn: School of Applied Science
Dean Harvey Brook
1 Copy

Commanding Officer and Director
U. S. Naval Training Device Center
Port Washington
Long Island, New York

Attn: Technical Library 1 Copy

Office of Naval Research
Washington 25, D. C.

Attn: Code 450, Dr. R. Trumbull
1 Copy

Wright Air Development Division
Electronic Technology Laboratory
Wright-Patterson AFB, Ohio

Attn: Lt. Col. L. M. Butsch, Jr
ASRUEB 1 Copy

Laboratory for Electronics, Inc
1079 Commonwealth Ave
Boston 15, Massachusetts

Attn: Dr. H. Fuller 1 Copy

Stanford Research Institute
Computer Laboratory
Menlo Park, California

Attn: H. D. Crane 1 Copy

General Electric Co.
Schnectady 5, N. Y.

Attn: Library, L.M.E. Dept
Bldg 28-501 1 Copy

The Rand Corp
1700 Main St
Santa Monica, California

Attn: Numerical Analysis Dept
Willis H. Ware 1 Copy

Massachusetts Institute of Technology
Cambridge 39, Massachusetts

Attn: Prof. John McCarthy
26-007B 1 Copy

DISTRIBUTION (Continued)

Office of Naval Research
Washington 25, D. C.

Attn: Code 430 — 1 Copy

Carnegie Institute of Technology
Pittsburgh, Pennsylvania

Attn: Director, Computation Center
Alan J. Perlis — 1 Copy

Rome Air Development Center, RCOR
DCS/ Operations, USAF
Griffiss Air Force Base, New York

Attn: Irving J. Gabelman
1 Copy

Air Force Office of Scientific Research
Directorate of Information Sciences
Washington 25, D. C.

Attn: Dr. Harold Wooster
1 Copy

Hunter College
New York 21, New York

Attn: Dean Mina Rees — 1 Copy

Radio Corporation of America
306/2
Data Systems Division
8500 Balboa Blvd
Van Nuys, California

Attn: Joseph E. Karroll
1 Copy

Mr. Sidney Kaplan
1814 Glen Park Avenue
Silver Spring, Maryland
1 Copy

Stanford Research Institute
Menlo Park, California

Attn: Dr. Charles Rosen
Applied Physics Laboratory
1 Copy

National Bureau of Standards
Washington 25, D. C.

Attn: Miss Ida Rhodes
220 Stucco Bldg — 1 Copy

L. G. Hanscom Field/AF-CRL-CRRB/
Bedford, Massachusetts

Attn: Dr. H. H. Zschirnt
1 Copy

Rome Air Development Center
Griffiss Air Force Base
Rome, New York

Attn: Mr. Alan Barnun — 1 Copy

Armour Research Foundation
10 West 35th Street
Chicago 16, Illinois

Attn: Mr. Scott Cameron
E. E. Research Dept
1 Copy

DISTRIBUTION (Continued)

Department of the Army
Office of the Chief of
Research and Development
Pentagon, Room 3D442
Washington 25, D. C.

Attn: Mr. L. H. Geiger
1 Copy

General Electric Research Lab
P. O. Box 1088
Schenectedy, New York

Attn: V. L. Newhouse
Applied Physics Section
1 Copy

Harvard Computation Lab
Harvard University
Cambridge, Massachusetts

Attn: Dr. Anthony Oettinger
1 Copy

National Bureau of Standards
Washington 25, D. C.

Attn: Mrs. Ethel Marden
1 Copy

Rand Corporation
1700 Main Street
Santa Monica, California

Attn: Library 1 Copy

University of Chicago
Committee on Mathematical Biology
Chicago, Illinois

Attn: Prof. H. D. Landahl
1 Copy

University of Pennsylvania
Moore School of Electrical Engrg
200 South 33rd Street
Philadelphia 4, Pennsylvania

Attn: Miss Anna Louise Campion
1 Copy

Department of the Army
Office of the Asst COFD
for Intellegence
Room 2B529, Pentagon
Washington, D. C.

Attn: John F. Kullgren 1 Copy

Mr. Robert F. Samson
Directorate of Intelligence
and Electronic Warfare
Griffiss Air Force Base
Rome, New York 1 Copy

Mr. Bernard M. Fry, Deputy Head
Office of Science Information Service
National Science Foundation
1951 Constitution Avenue, N. W.
Washington 25, D. C. 1 Copy

International Business Machines, Corp
Advanced Systems Development Div
San Jose 14, California

Attn: I. A. Warheit 1 Copy

Harry Kesten
Cornell University
Dept of Mathematics
Ithaca, New York 1 Copy

DISTRIBUTION (Continued)

Applied Physics Laboratory
Johns Hopkins University
8621 Georgia Avenue
Silver Spring, Maryland

Attn: Document Library — 1 Copy

Chief, Bureau of Supplies and Accounts
Navy Department
Washington, D. C.

Attn: Code W3 — 1 Copy

Bendix Products Division
Bendix Aviation Corporation
Southbend 20, Indiana

Attn: E. H. Crisler — 1 Copy

Officer in Charge
U. S. Naval Photographic
Interpretation Center
4301 Suitland Road
Suitland, Maryland

Attn: Mr. J. Pickup — 1 Copy

Dr. Noah S. Prywes
Moore School of Engineering
University of Pennsylvania
Philadelphia 4, Pennsylvania — 1 Copy

Auerbach Electronics Corporation
1634 Arch Street
Philadelphia 3, Pennsylvania — 1 Copy

National Security Agency
Fort George G. Meade
Maryland

Attn: R.-42, R. Wiggington — 1 Copy

Peter H. Greene
Committee on Mathematical Biology
University of Chicago
Chicago, Illinois — 1 Copy

Federal Aviation Agency
Bureau of Research and
Development
Washington 25, D. C.

Attn: RD-375/Mr. Harry Hayman — 1 Copy

National Security Agency
Fort George G. Meade
Maryland

Attn: Librarian, C-332 — 1 Copy

Federal Aviation Agency
Bureau of Research and
Development Center
Atlantic City, New Jersey

Attn: Simon Justman — 1 Copy

DISTRIBUTION (Continued)

Cornell Aeronautical Laboratory, Inc
P. O. Box 235
Buffalo 21, New York

Attn: Systems Requirements Dept
A. E. Murray 1 Copy

Institute for Space Studies
475 Riverside Drive
New York 27, New York

Attn: Mr. Albert Arking
1 Copy

Chief, Bureau of Ships
Code 671A2
Washington, D. C.

Attn: LCdr. E. B. Mahinske, USN
1 Copy

Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

Attn: Library 1 Copy

Electronics Research Laboratory
University of California
Berkeley 4, California

Attn: Director 1 Copy

Mr. Gordon Stanley, No. 58
Defense Research Laboratories
General Motors Corp
Box T
Santa Barbara, California
1 Copy

A. J. Cote, Jr.
Applied Physics Laboratory, JHU
The Johns Hopkins University
8621 Georgia Avenue
Silver Spring, Maryland
1 Copy

Institute for Defense Analysis
Communications Research Division
Von Neumann Hall
Princeton, New Jersey
1 Copy

Mr. John Cook Wyllie
Alderman Library
University of Virginia
Charlottesville, Virginia
1 Copy

Julian J. Bussgang
Signatron, Inc.
Miller Building
594 Marrett Road
Lexington 73, Massachusetts
1 Copy

Air Force Office of Scientific Research
Information Research Div
Washington 25, D. C.

Attn: R. W. Swanson 1 Copy

Rome Air Development Center
Griffiss Air Force Base
Rome, New York

Attn: Mr. Fred Dion 1 Copy

Office of Research and Development
U. S. Patent Office
Washington 25, D. C.

Attn: The Librarian 1 Copy